VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Predikce meteorologických jevů pomocí neuronových sítí

Meteorological Phenomenon Prediction Using Neural Networks

2013 Bc. Lukáš Vitásek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Lukáš Vitásek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Predikce meteorologických jevů pomocí neuronových sítí
Meteorological Phenomenon Prediction Using Neural Networks

Zásady pro vypracování:

Diplomant se bude v práci zabývat predikcí meteorologických jevů - krátkodobých předpovědí srážek - na základě radarových dat.
Práce bude obsahovat:
1. Shrnutí současného stavu poznání.
2. Implementace vybraného typu umělé neuronové sítě.
3. Experimenty s implementovanou neuronovou sítí.
4. Vyhodnocení experimentů.

Seznam doporučené odborné literatury:

[1] Heaton Jeff. Introduction to the Math of Neural Networks, Heaton Research, 2012
[2] Šíma, Jiří a Roman Neruda. Teoretické otázky neuronových sítí. Vyd. 1. Praha: Matfyzpress, 1996. ISBN 80-85863-18-9

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Doc. Mgr. Jiří Dvorský, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2013 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 .....

Rád bych touto cestou poděkoval vedoucímu diplomové práce doc. Mgr. Jiřímu Dvorskému, Ph.D. a Ing. Janu Martinoviči, Ph.D. za jejich cenné rady, připomínky a trpělivost při vedení mé diplomové práce.

Abstrakt

Tato práce se zaobírá krátkodobými předpověďmi srážek pomocí neuronových sítí. Námi navržená metoda pak za účelem předpovědi budoucího vývoje srážkové situace používá pouze radarové snímky poskytované ČHMÚ ze sítě radarů CZRAD. K naučení neuronové sítě je také vyzkoušeno několik metod a jejich modifikací spadajících do kategorie evolučních algoritmů, konkrétně genetické algoritmy, diferenciální evoluce a SOMA.

Klíčová slova: umělé neuronové sítě, evoluční algoritmy, genetický algoritmus, diferenciální evoluce, krátkodobé předpovědi srážek, radar pro počasí

Abstract

This thesis deals with the problem of short term rainfall prediction using neural networks. The method developed by us uses only radar images provided by ČHMÚ from CZRAD radar network to predict precipitation development. Genetic algorithms, differential evolution and SOMA, the methods that belongs to the category of evolutionary algorithms, have been tested to learn neural network.

Keywords: artificial neural networks, evolutionary algorithms, genetic lagorithm, differential evolution, percipitation nowcasting, weather radar

Seznam použitých zkratek a symbolů

ANN	–	artificial neural network
ARMA	–	auto-regressive moving-average model
BMA	–	Bangkok Metropolitan Administration
BP	–	back propagation
DE	–	diferential evolution
FFNN	–	feedforward neural network
GA	–	genetic algorithm
KNN	–	n-nearest neighbors algorithm
MLP	–	multilayer perceptron
RWS	–	roulette wheel selection
SUS	–	stochastic universal selection, stochastic universal sampling
TMD	–	Thai Meteorological Department

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Přesná předpověď počasí je důležitá pro mnoho odvětví lidských činností. Příklady použití nalezneme například při plánování v zemědělství nebo řešení krizových situací, jako například povodně. V průběhu takovéto mimořádné události je na dostupnost co nejpřesnějších informací kladem mimořádný důraz. Díky nim totiž se lze s předstihem připravit na budoucí vývoj situace a minimalizovat či úplně eliminovat možné následky.

Cennými zdroji dat při těchto předpovědích jsou jednak pozemní stanice, zaznamenávající například aktuální teplotu, vlhkost vzduchu nebo úhrn spadlých srážek, tak i různé satelitní či radarové snímky. Pro Českou republiku jsou například dostupné aktuální radarové snímky[1], poskytované ČHMÚ díky radarové síti CZRAD.

S postupem času bylo vyvinuto či použito několik metod [6] vhodných pro predikci počasí. Tato práce si klade za úkol prozkoumat jednu z nich, použití neuronových sítí, která už byla několikrát úspěšně použita [5, 6, 7]. Jako metody učení bylo vybráno několik algoritmů spadajících do kategorie evolučních algoritmů.

Pokud bude námi prezentovaný přístup fungovat dobře, rádi bychom našli jeho uplatnění například v projektu *Floreon*$^+$, jehož cílem[2] je monitorování, modelování, predikce a podpora řešení krizových situací, především pro oblast Moravskoslezského kraje. Kromě hlavního zaměření, hydrologie, se také zabývá modelováním znečištění životního prostředí či monitorováním a predikcí dopravních situací.

[1] http://portal.chmi.cz/files/portal/docs/meteo/rad/data_jsradview.html
[2] http://floreon.vsb.cz

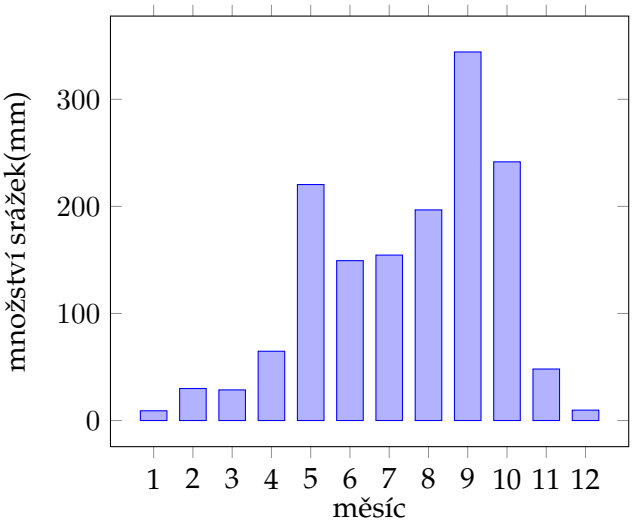
2 Aplikace neuronových sítí

2.1 Bangkok

Cílem studie [5] bylo zlepšit přesnost dosavadní metody pro predikci množství srážek v reálném čase pro město Bangkok v Thajsku. Tato metropole, ležící na deltě řeky Chao Phraya a rozkládající se na celkové ploše $1569\,\mathrm{km}^2$, je často sužována povodněmi ovlivňovaných několika faktory. Jednak se jedná o postupný propad půdy v rozmezí $10\,\mathrm{mm}$ až $30\,\mathrm{mm}$ za rok spolu s už tak nízkou nadmořskou výškou $1.5\,\mathrm{m}$, ale také nedostatečný odvodňovací mechanismus vyplývající z rychlé urbanizace. Příčinou samotných povodní je jednak nárůst hladiny způsobený náhlým zvýšením množství vodní masy od míst ležících proti proudu řeky, ale také deště zasahujícího přímo město. Roční souhrn srážek činí přibližně[3] $1500\,\mathrm{mm}$, ke kterým přispívají monzuny od května do října, zbytek roku je naopak převážně suchý s občasnými bouřkami.

Měsíc	Srážky(mm)
leden	9.1
únor	29.9
březen	28.6
duben	64.7
květen	220.4
červen	149.3
červenec	154.5
srpen	196.7
září	344.2
říjen	241.6
listopad	48.1
prosinec	9.7

Tabulka 1: Průměrné množství srážek v Bangkoku za období 1961-1990



Obrázek 1: Průměrné množství srážek v Bangkoku za období 1961-1990

Ochranu města před povodněmi zajišťuje *BMA* několika programy. Mezi ně patří například systém odvodňovacích kanálů se senzory pro získávání aktuálních hodnot o výšce vodní hladiny, ale také celkem 53 stanic pro měření srážek rozmístěných po celém městě. Oba systémy jsou schopné podávat aktuální informace o stavu v intervalech s rozestupem 15 minut. Mimo to je Bangkok a jeho blízké okolí pokryto dalšími 51 měřícími stanicemi patřícími *TMD*. Stanice patřící oběma oddělením jsou schopna měřit data s přesností na $0.5\,\mathrm{mm}$.

Pro účely studie bylo z tohoto celkového počtu 104 stanic vybráno pouze 75 z nich, každá zabírající území o rozloze přibližně $21\,\mathrm{km}^2$. Důvodem pro vyloučení některých

[3] http://www.tmd.go.th/en/province_stat.php

z nich se stala jejich poloha, neležely totiž přímo v Bangkoku, ale v jeho okolí. Po následné analýze dat se stanovilo časové rozmezí od 1.1. 1997 do 31.12. 1999 jako vhodné k učení neuronových sítí, data za celý rok 2003 posloužila jako ověřovací množina.

Následovalo porovnání několika modelů vyvinutých pro predikci lišících se v použitém typu neuronové sítě(*MLP*, *FFNN*), počtu neuronů v jednotlivých vrstvách, ale také ve vstupních datech. Všechny zkoumané modely používaly strukturu sítě o dvou skrytých vrstvách a jediném výstupním neuronu určujícím hodnotu srážek na daném místě pro aktuální čas. K naučení těchto neuronových sítí byly použita data zahrnující kromě hodnot srážek také další meteorologické parametry jako relativní vlhkost, tlak vzduchu, teplota mokrého teploměru a oblačnost.

Z prezentovaných výsledků projevoval největší přesnost model s *FFNN*, který používá hyperbolickou tangens jako aktivační funkci a na jehož vstupy jsou přiváděny hodnoty:

$$R_t, RH_t, WBT_t, AP_t, CL_t, AR_t, SR1_t, SR2_t, SR3_t$$

kde
- R_t je intenzita srážek na dané stanici v čase t
- RH_t je relativní vlhkost v čase t
- WBT_t je teplota mokrého teploměru v čase t
- AP_t je tlak vzduchu v čase t
- CL_t je oblačnost v čase t
- AR_t je průměrná hodinová intenzita srážek na všech stanicích v čase t
- $SR1_t$ je intenzita srážek na okolní stanici 1 v čase t

2.2 Irsko

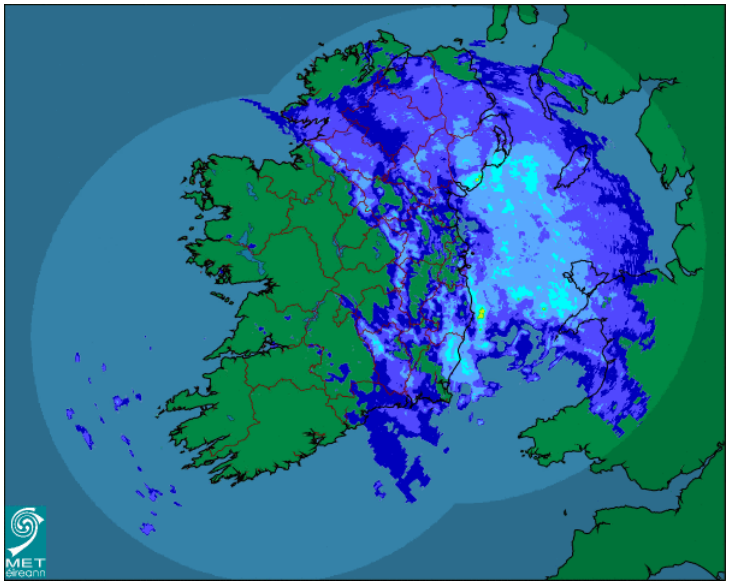
Tato práce [7] se zabývala vývojem systému pro krátkodobé předpovědi srážek nad územím Irska. Celková rozloha území byla oproti předchozí studii několikanásobně větší, protože data poskytovaná radary Met Éireann[4], ukázkový snímek je na obrázku 2, zachycují plochu o celkové rozloze $250\,000\,\mathrm{km}^2$.

V současné době jsou v provozu 2 takovéto radary. První stanice je umístěna na letišti v Dublinu[5], přibližně $10\,\mathrm{km}$ od centra města, druhá se nachází na letišti v Shannonu[6]. Obě dvě jsou schopny dodávat aktuální snímky v patnáctiminutových intervalech s dosahem $240\,\mathrm{km}$ a rozlišením $1\,\mathrm{km}$.

Samotný proces předpovědi probíhal následovně:

1. získání aktuálních snímků
2. zpracování snímků
   (a) odečtení pozadí ze snímků
   (b) binarizace

[4] http://www.met.ie/latest/rainfall_radar.asp
[5] http://www.met.ie/aviation/dublin.asp
[6] http://www.met.ie/aviation/shannon.asp



Obrázek 2: Snímek poskytovaný Met Éireann

(c) odstranění regionů které nebyly vzhledem k předpovědi potřebné a také malých shluků pixelů

(d) identifikace srážek

(e) identifikace hranic srážek pomocí detekce hran

3. extrakce srážkových dat

pro účely dalšího zpracování dat byla masa srážek popsána n-ticí

$$R_m = \{P_{hy}; M_{or}\}$$

kde P_{hy} reprezentuje fyzikální parametry
M_{or} reprezentuje morfologické parametry

Oba tyto parametry lze dále rozdělit na

$$P_{hy} = \{L_{CoG}; V_{el}; D_{ir}; A_{cel}; T\}$$

$$M_{or} = \{C_{on}; A_{rea}; I; C_{omp}; C_{lu}\}$$

kde L_{CoG} je pozice těžiště vodní masy
V_{el} je rychlost
D_{ir} je směr
A_{cel} je zrychlení
T je časové razítko
C_{on} je množina bodů reprezentující obrys masy
A_{rea} je celková plocha
I je intenzita srážky
C_{omp} je kompaktnost srážkové masy
C_{lu} je charakteristika shlukování

4. predikce vývoje srážek

Pro ověření přesnosti předpovědi pomocí vyvinuté metody bylo vybralo 14 různých míst, pro které byly vypočítány následující hodnoty:

$$Precision = \frac{R/R}{R/R + R/NR} \tag{1}$$

$$Recall = \frac{R/R}{R/R + NR/R} \tag{2}$$

kde R/R je počet úspěšných předpovědí kde opravdu pršelo
R/NR je počet předpovědí kde mělo pršet ale nepršelo
NR/R je počet predikcí bez deště kde ve skutečnosti pršelo

Pomocí těchto dvou vztahů se vypočítá tzv. $F\text{–}Score$ jako:

$$F\text{–}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

Z prezentovaných výsledků bylo zřejmé, že nejlepší skóre získaly místa vzdálená přibližně 100 km od radaru, nejhorší naopak místa nejvíce vzdálená od obou radarů. Samotná města Dublin a Shannon neměla dobré výsledky, které ale nebyly tak špatné jako u vzdálených míst. Důvodem bylo, že tato místa leží příliš blízko samotných radarů.

2.3 Apeniny, řeka Sieve

Článek [6] se zaobírá zkoumáním přesnosti modelu *ARMA*, neuronových sítí a neparametrické metody nejbližších sousedů pro krátkodobé předpovědi srážek v rozsahu 1 až 5 hodin v povodí řeky Sieve. Pro účely předpovědí byly k dispozici data o jejím hodinovém průtoku za období 1.1.1992 až 31.12.1996. Za stejné období byly také dostupná data o aktuální teplotě na 4 stanicích a údaje z 12 srážkoměrů.

Protože tyto data obsahovala převážně nerelevantní údaje, jako například období bez srážek, byly vybrány pouze bouřkové události. Jejich definice požadovala, aby během jedné hodiny napadlo více než 1 mm srážek, tomu muselo předcházet minimálně 5 a následovat alespoň 20 hodin srážek menších než 1 mm, přičemž průtok řekou musel po celou dobu stoupat o $20\,\mathrm{m^3\,h^{-1}}$.

Při vývoji metody používající neuronové sítě se testovalo několik variant *BP*, ze kterých dokázala nejrychleji naučit neuronovou síť, a také méně pravděpodobněji uvíznout v okolí lokálního minima, metoda *Levenberg-Marquardt*.

Autoři také otestovali dva typy předpovědí časových událostí s krokem vetším než 1. První metoda[7] spočívala v použití neuronové sítě, která předpoví pouze jeden následující krok. Výsledek předpovědi se pak zvonu použije v dalším časovém úseku na vstupu. Druhá[8] oproti tomu používala neuronovou síť, jejíž výstupy přímo udávaly hodnotu pro několik časových úseků dopředu. První zmíněná pak prokazovala nejlepší výsledky pokud byla použita pouze pro předpověď jednoho budoucího kroku. Při rekurzivním použití se propagovala chyba jednotlivých předpovědí do dalších iterací, proto byla pro použití v předpovědi na více časových kroků zvolena druhá metoda.

V závěru autoři porovnali úspěšnost různých variant výše zmíněných modelů. V předpovědích pro jeden časový krok dopředu (1 h) projevoval nejlepší úspěšnost *adaptivní ARMA* model. S narůstajícím počtem časových kroků ale jeho přesnost klesala nejvíce ze všech testovaných modelů. Korelační koeficient při předpovědi na 6 h činil pouze 0.003. Naproti tomu, varianta používající neuronovou síť učenou metodou *split-sample*[9] si při tomto časovém kroku dokázala udržet korelační koeficient na hodnotě 0.372, druhého nejlepšího výsledku (korelační koeficient 0.162) dosáhla také neuronová síť, na rozdíl od předchozí využívající adaptivní učení.

[7] recursive multi-step

[8] direct multi-step

[9] Data byla rozdělena na trénovací a testovací množinu, kdy trénovací množina obsahovala dvojnásobné množství událostí oproti testovací.

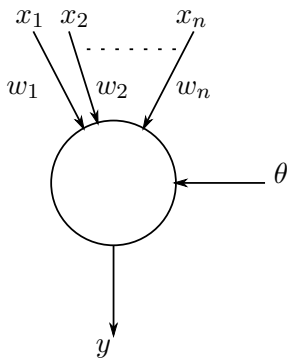
3 Neuronové sítě

Mozek je orgán tvořící centrum nervového systému obratlovců, který se na buněčné úrovni skládá ze dvou typů buněk, neuronů a glií. Pokud bychom chtěli hovořit o konkrétních počtech[10], tak lidský mozek tvoří okolo $1.7 * 10^{11}$ buněk, z čehož více než polovinu ($8.6 * 10^{10}$ až 10^{11}) reprezentují neurony, propojené navzájem okolo 10^{14} synapsemi. Neurony také zastávají primární činnosti, přenos a zpracování informace. Glie naproti tomu slouží jako podpůrný typ buňky pro samotné neurony, který je obklopuje, odděluje a udržuje na místě, dodává kyslík nebo odstraňuje odumřelé.

Samotný přenos informace mezi neurony je realizován kombinací elektrických a chemických procesů. K tomuto účelu má tělo neuronu jednak maximálně jedno axonové vlákno, kterým se propaguje elektrický signál dále k ostatním, ale také zpravidla i více než jeden dendrit, postupně se větvící přes který je signál přijímán od ostatních. Propojení mezi zakončením axonového vlákna a dendrit ale není realizováno přímo. Mezi nimi se nachází synaptická štěrbina, která přeměňuje elektrické vzruchy na chemické látky. Pokud jejich množství překročí určité množství, vytvoří se nový vzruch šířící se dále. Mez přitom může být dosažena sečtením vzruchů více neuronů naráz, případně od jednoho posílajícího vzruchy v rychlých časových intervalech.

3.1 Umělé neuronové sítě

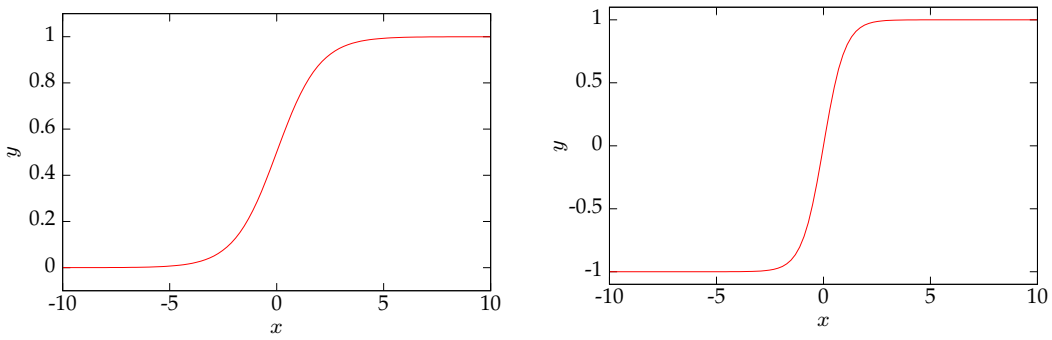
Umělé neuronové sítě [2, 12] jsou velice zjednodušeným modelem skutečných. Základní jednotku zde tvoří neuron, mající n vstupů x_1 až x_n, přičemž ke každému vstupu náleží vlastní váha w_i, která může zesílit, či zeslabit jí procházející signál. Takto upravené vstupy dále podléhají sumaci, od které se odečte prahová hodnota θ. Získaný výsledek se použije jako vstup pro aktivační funkci f, jejíž hodnota reprezentuje výstup neuronu y.



Obrázek 3: Umělý neuron

Takto definovanou jednotku lze matematicky vyjádřit jako:

[10] http://blogs.scientificamerican.com/brainwaves/2012/06/13/know-your-neurons-what-is-the-ratio-of-glia-to-neurons-in-the-brain/



(a) Sigmoida: $S(x) = \frac{1}{e^{-x}}$

(b) Hyperbolický tangens: tanh $x = \frac{e^{2x}-1}{e^{2x}+1}$

Obrázek 4: Aktivační funkce běžně používané u neuronových sítí

$$y = f\Big(\sum_{i=1}^{n} w_i x_i - \theta\Big) \tag{4}$$

Prahovou hodnotu θ zde můžeme zaměnit za další vstup neuronu x_0 s hodnotou excitace $x_0 = 1$ a váhou w_0 jemu příslušící $w_0 = -\theta$. Tímto se předchozí funkce zjednoduší na:

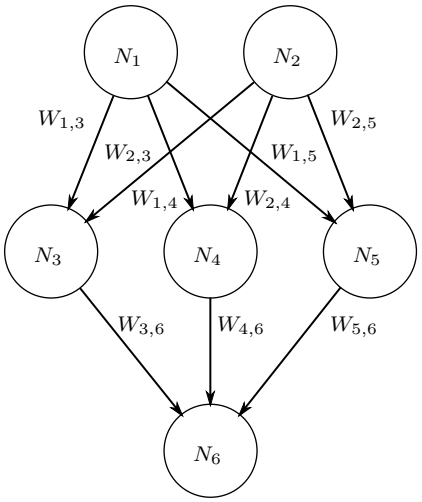
$$y = f\Big(\sum_{i=0}^{n} w_i x_i\Big) \tag{5}$$

Různé druhy neuronových sítí se pak odlišují použitými aktivačními funkcemi, pravidly aplikovanými při tvorbě struktury sítě pro jednotlivá propojení.

Podle topologie pak můžeme sítě rozdělit na dva základní typy, s pouze dopřednými vazbami a sítě povolující také rekurentní vazby. Struktura dopředných sítí je tvořena jednou vstupní vrstvou, jednou výstupní vrstvou, a s libovolným počtem tzv. skrytých vrstev (nebo nemusí být přítomny vůbec). Neurony v jednotlivých vrstvách nejsou navzájem propojeny, výstupy neuronů z předchozí vrstvy jsou propojeny se vstupy následující vrstvy.

Podle způsobu jakým jsou neuronové sítě učeny lze strategie učení rozdělit na dvě skupiny. U první z nich, **učení bez učitele**, si neuronová síť musí v předkládaných vstupních datech sama nalézt vzory. Této definici odpovídají následující:

- self-organizing map (SOM)
- neural gas
- adaptive resonance theory (ART)
- Learning Vector Quantization (LVQ)



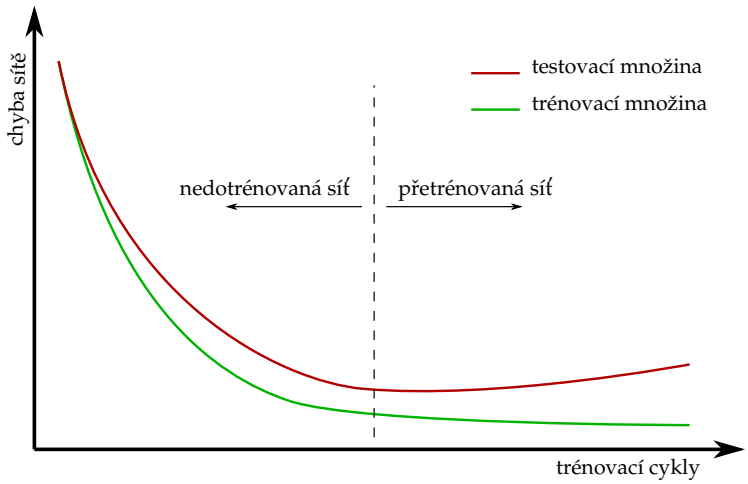
Obrázek 5: Vrstvená neuronová síť

Protipól k předchozím metodám tvoří **učení s učitelem**, kde jsou navíc k předkládaným vstupním datům dostupné také očekávaná výstupní data. Neuronová síť je tedy v průběhu učení upravovaná na základě chyby se kterou odpovídá na konkrétní data.

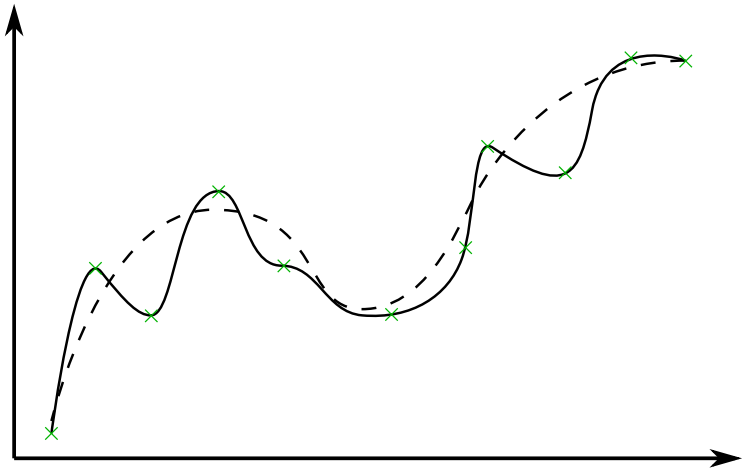
- hebb rule
- back-propagation

Při procesu učení může nastat problém, že síť naučíme na trénovací data tak dobře, že se zhorší její generalizace pro data, která jí prozatím nebyla prezentovaná. Ty totiž často mohou pocházet z měření u kterého není možné vyloučit vliv rušení a síť se tímto může naučit i tuto chybu. Aby se takovéto situaci předešlo, dělí se množina dostupných dat na tři části, *trénovací množinu*, *testovací množinu* a *validační množinu*.

Trénovací množina dat, jak již název napovídá, se použije při samotném procesu učení k výpočtu chyby sítě pro konkrétní vzory, na jejíž základě se upravuje konfigurace sítě. Zároveň se po určitém počtu trénovacích cyklů provede výpočet chyby naučené sítě pomocí testovací množiny dat. V případě, že obě hodnoty klesají, učení probíhá dále. Pokud naopak začne stoupat chyba vypočtená z testovacích dat, je potřeba vrátit neuronovou síť do předchozí konfigurace a proces trénování tímto končí. Za takto nalezeným bodem bychom získali přeučenou síť.



Obrázek 6: Průběh učení sítě



Obrázek 7: Přetrénování neuronové sítě

4 Evoluční algoritmy

4.1 Genetické algoritmy

Genetické algoritmy jsou heuristická metoda a globální optimalizační technika snažící se napodobit Darwinovu teorii evoluce při hledání řešení problémů.

První ne příliš úspěšné počátky s aplikací evoluce lze datovat do 50. let 20. století, kdy tyto metody používaly pouze metody mutace genetického materiálu. V 60.letech přišel *Hans J. Bremermann* [11] s myšlenkou použití reprodukce pomocí křížení, tehdy ještě založenou na sumaci odpovídajících částí genů. Následný pokrok přišel díky přesvědčení *Johna H. Hollanda* [8], že rekombinace genů pomocí páření je nezbytnou částí evoluce. Prvním výsledkem jeho práce byl klasifikátor pracující s množinou pravidel a binárními řetězci, kde jednotlivé bity reprezentovaly (ne)splnění konkrétní podmínky a pravidla určovala akce prováděné po splnění požadovaných podmínek. Později také přišel s teorémem schémat, který dnes paří mezi teorie konvergence genetických algoritmů.

Tak jako samotný princip evoluce, i terminologii si genetické algoritmy vypůjčují z přírody. Algoritmus pracuje s *populací* obsahující množinu *jedinců* reprezentujících možná řešení problému. Každého jedince lze vyjádřit souborem parametrů, zde nazvaných *geny*, tvořící dohromady jediný *chromozom*. Úspěšnost jednotlivých řešení je vyjádřena odpovídající hodnotou *fitness*. Ta slouží jako rozhodovací kritérium pro přežití do další generace nebo účast na reprodukci. Tak jako v přírodě, i zde totiž platí zákon přežití nejsilnějších (nejlépe adaptovaných).

Pokud bychom toto měli vyjádřit na nějakém konkrétním příkladu, můžeme například hledat řešení pro rovnici polynomu $f(x) = a_0 + a_1x + a_2x^2 \dots a_nx^n$ tak, aby procházel předem zadanými body. Hodnota *fitness* tak může představovat sumu vzdáleností mezi očekávanou hodnotou y_i každého bodu s aktuální hodnotou $y_{actual} = f(x_i)$ funkce v daném bodě. Samotný chromozom bude reprezentován koeficienty polynomu $a_0 \dots a_n$ (geny).

Samotný evoluční cyklus lze rozdělit do dvou na sebe navazujících částí. Jako první probíhá proces selekce jedinců, který má za úkol vybrat z aktuální populace ty jedince, kteří z pohledu fitness funkce obsahují ve svém genetickém kódu lepší řešení. Tito vybraní jedinci mají následně možnost se podílet na tvorbě nových potomků pomocí procesu reprodukce, který lze rozdělit do dvou částí. Jednak se jedná o křížení, dále pak mutace.

4.1.1 Vhodnost

Aby bylo možné nějak rozlišit úspěšnost jednotlivých řešení mezi sebou, je potřeba vypočítat jejich *účelovou funkci* (*cost function*). Ta může reprezentovat například hodnotu optimalizované funkce v daném bodě, délku cesty při řešení problémů typu obchodního cestujícího, cenu výrobku. Samotný algoritmus při evoluci následně používá takzvanou *vhodnost* (*fitness function*), což může být přímo hodnota účelové funkce, její normalizovaná hodnota nebo transformovaná hodnota do intervalu $\langle 0; 1 \rangle$.

[11] http://berkeley.edu/news/media/releases/96legacy/releases.96/14319.html

Input: $Population_{size}$, $Dimension$, $Generations$, $P_{mutation}$, $P_{ccossover}$
Output: S_{best}
$Population$ = CreateInitialPopulation($Population_{size}$, $Dimension$);
Evaluate($Population$);
S_{best} = FindBestSolution($Population$);
for $i = 0$ ***to*** $Generations$ **do**
    $Parents$ = SelectParents($Population$);
    $Offspring = \emptyset$;
    foreach $Parent_1$, $Parent_2$ ***in*** $Parents$ **do**
        $Child$ = Crossover($Parent_1$, $Parent_2$, $P_{crossover}$);
        Mutate($Child$, $P_{muattion}$);
        Evaluate($Child$);
        $Offspring$ += $Child$;
    end
    $Population$ = GetNewPopulation($Population$, $Offspring$);
    S_{best} = FindBestSolution($Population$);
end

Algorithm 1: Genetický algoritmus

Obvykle jsou algoritmy řešeny pro hledání maxima na vhodnosti. Pokud hledáme minimum funkce, je potřeba vhodně převést účelovou funkci na vhodnost. K tomuto účelu lze použít následujícího převodu [4], který minimální hodnotě účelové funkce přidělí maximální hodnotu vhodnosti a naopak, zbytek hodnot bude lineárně přerozdělen mezi těmito vzniklými hraničními body.

$$f_i = \frac{f_{max} - f_{min}}{c_{min} - c_{max}} * c_i + \frac{c_{min} f_{min} - c_{max} f_{max}}{c_{min} - c_{max}} \tag{6}$$

kde
- c_{max} je maximální hodnota účelové funkce
- c_{min} je minimální hodnota účelové funkce
- f_{max} je maximální hodnota vhodnosti
- f_{min} je minimální hodnota vhodnosti
- c_i je hodnota účelové funkce pro daného jedince
- f_i je hodnota vhodnosti pro daného jedince

Pro převod přímo do intervalu $\langle 0; 1 \rangle$ lze rovnici upravit na:

$$f_i = \frac{1}{c_{min} - c_{max}} [(1 - \epsilon) c_i + c_{min} \epsilon - c_{max}] \tag{7}$$

- ϵ je malé kladné číslo, např. 0.01

4.1.2 Selekce

Selekce jedinců účastnících se na reprodukci je prováděna na základě vhodnosti každého jedince většinou pomocí tzv. *výběru ruletou* (*roulette wheel selection*, *RWS*). Implementace algoritmu je přímočará, počítá pouze s faktem, že se hledá maximum na fitness funkci.

Před prvním výběrem v každé generaci je potřeba přepočítat pravděpodobnosti výběru jedinců následujícím postupem:

1. vypočtení fitness celé populace

$$F = \sum_{i=1}^{PopSize} f_i$$

2. vypočtení pravděpodobnosti selekce pro každého jedince

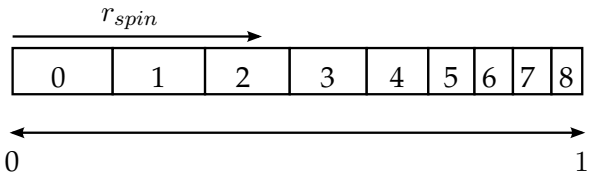
$$p_i = \frac{f_i}{F}$$

3. vypočtení kumulativní pravděpodobnosti selekce pro každého jedince

$$q_i = \sum_{j=1}^{i} p_j$$

Samotný výběr pak probíhá vždy tak, že je vygenerováno náhodné číslo $r_{spin} \in \langle 0; 1 \rangle$, reprezentující pootočení na ruletě, a hledá se jedinec pro něhož platí, že:

$$q_{i-1} < r_{spin} \leq q_i$$



Obrázek 8: RWS

Nejjednodušší implementace může používat lineární vyhledávání, které má složitost výběru n^2. Pokud si ale uvědomíme fakt, že hledáme číslo ve vzestupně seřazeném seznamu, použitím binárního vyhledávání klesne složitost na $n \log(n)$.

Při použití *RWS* ale může nastat situace, že náhodou méně vhodný jedinec dostane více šancí k reprodukci než více vhodný. Přístup snažící se eliminovat tuto nevýhodu se jmenuje *stochastický univerzální výběr* (*stochastic universal selection*, *stochastic universal sampling*, *SUS*). Zde se také provádí výběr podle rulety, nicméně je mírně upravený. Při každém výběru se pokaždé nevychází z jednoho počátečního bodu rulety, ale těchto

počátků je více. Přesněji tolik, kolik jedinců se vybírá z aktuální populace, navíc jsou na ruletě rozmístěny s rovnoměrnými odstupy mezi sebou.

Postup před započetím výběru je téměř stejný jako u *RWS*, jsou pouze přidány dva kroky:

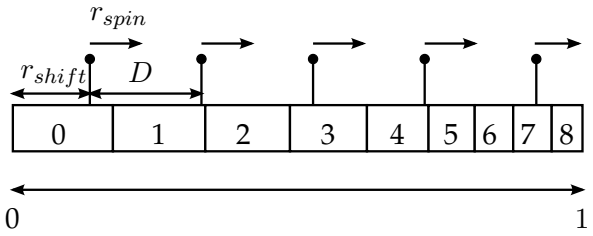
1. vypočtení rozestupu mezi ukazateli

$$D = \frac{1}{N}$$

2. získání posunutí prvního ukazatele

$$r_{shift} \in \langle 0; D \rangle$$

Selekce jedinců pak probíhá tak, že pro každou generaci se vygeneruje pouze jedno náhodné číslo $r_{shift} \in \langle 0; 1 \rangle$ a od každého ukazatele se provede výběr. Jako výhodu získáme menší výpočetní zátěž, protože odpadá $N - 1$ generování náhodných čísel (po-otočení rulety).



Obrázek 9: SUS

Postup, kterým jsou rodiče vybíráni, může být i kombinací více přístupů. Například algoritmus v [1] používá k výběru prvního rodiče *RWS*, druhý je vybrán z populace náhodně.

Při výběru rodičů může nastat situace, že se v populaci vyskytne několik jedinců majících o tolik lepší hodnotu fitness, že obdrží nepřiměřeně mnoho možností k produkci potomků. Toto by nebyl problém v situaci, pokud takovíto jedinci obsahují ve svém kódu řešení, díky kterému by algoritmus směřoval ke globálnímu extrému. V počátečních fázích evoluce, kdy existuje ještě velká populační rozmanitost a prozkoumává se tedy velká oblast řešení, bývají takovéto extrémy pouze předzvěstí některého lokálního minima.

Přístupů jak vyřešit tuto situaci je několik. Pro výběr rodičů je v [9] uvedeno rozdělení na dva typy. První, nazvaná *explicitní přemapování fitness*, se snaží vyřešit situaci upravením distribuce samotné fitness hodnoty. Druhá metoda, nazvaná *implicitní přemapování fitness*, se naproti tomu snaží vyhnout procesu přemapování fitness hodnot úplně. Jako metoda dominující mezi ostatními byla zmíněna *pořadová selekce*.

Explicitní přemapování fitness

Pořadová selekce (*rank selection*, *fitness ranking*) je metoda snažící se obejít rozdíly mezi fitness hodnotami tím, že fitness je odvozena od pořadí každého jedince v populaci seřazené podle hodnoty účelové funkce.

První varianta *lineárního pořadového přerozdělení* přidělí hodnotu fitness přímo pořadí jedince v populaci:

$$s_i = rank_i \tag{8}$$

K tomuto existuje i zobecněná varianta umožňující upravit selekční tlak (selection pressure) pomocí sklonu křivky přerozdělení:

$$s_i = 2 - s_{pressure} + \left(2 * (s_{pressure} - 1) \frac{rank_i - 1}{pop_{size} - 1} \right) \tag{9}$$

kde $s_{pressure}$ je selekční tlak $s_{pressure} \langle 1.0; 2.0 \rangle$
pop_{size} je velikost populace
exponential

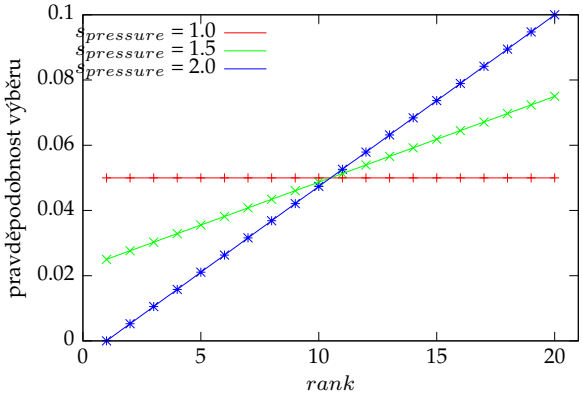
$$s_i = m^{(rank_i - 1)} \tag{10}$$

V případě kdy je potřeba vyvinout větší selekční tlak směrem k lepším řešením, lze použít *nelineární pořadové přerozdělení*. Distribuci pravděpodobností selekce jedinců lze ovlivnit parametrem c, který může být po celou dobu evoluce konstantní.

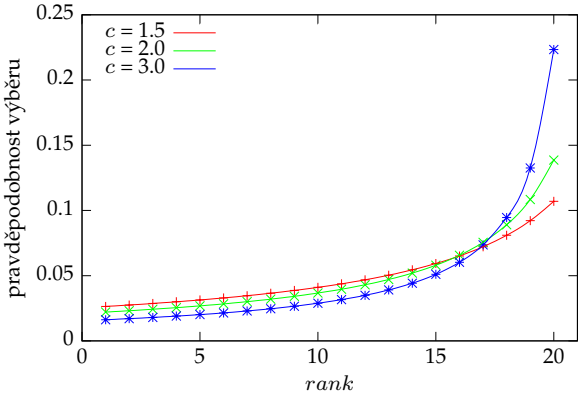
$$s_i = \frac{1}{c} ln \left(\frac{pop_{size} - (rank_i - 1)(1 - \frac{1}{e^c})}{pop_{size} - rank_i (1 - \frac{1}{e^c})} \right) \tag{11}$$

Selekce ořízntutím (*top selection, truncation selection*) přistupuje k řešení jinak. Rozděluje populaci na dvě části tak, že jedinci s pořadím nižším než $rank_{min}$ přijdou o možnost se účastnit reprodukce, zbylým jedincům se přidělí stejné šance.

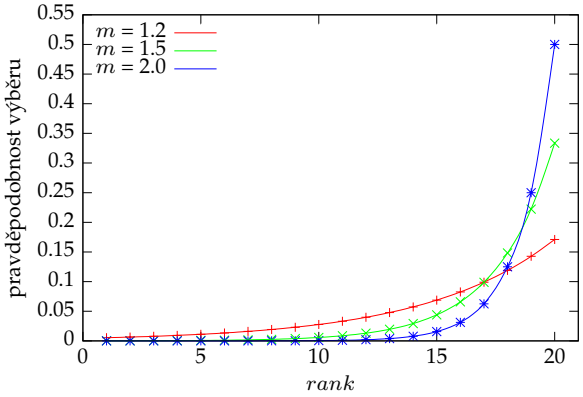
$$s_i = \begin{cases} \frac{1}{pop_{size} - rank_i + 1} & \text{pokud } rank_i \geq rank_{min} \\ 0 & \text{pokud } rank_i < rank_{min} \end{cases} \tag{12}$$



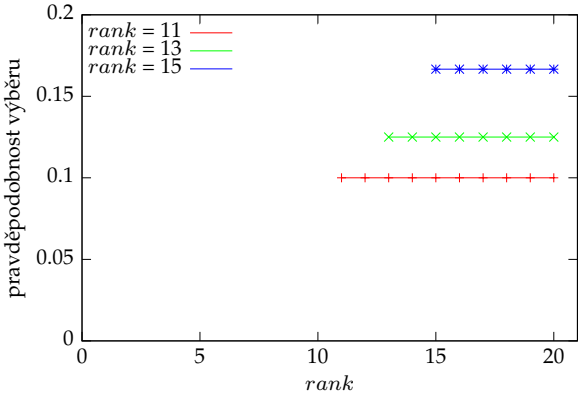
(a) Lineární přerozdělení podle rovnice 9 pro různé hodnoty parametru $s_{pressure}$



(b) Nelineární přerozdělení podle rovnice 11 pro různé hodnoty parametru c



(c) Exponenciální přerozdělení podle rovnice 10 pro různé hodnoty parametru c



(d) Selekce oříznutím podle rovnice 12 pro různé hodnoty parametru $rank_{min}$

Obrázek 10: Porovnání různých typů přerozdělení podle pořadové selekce

Implicitní přemapování fitness

Do této kategorie patří různé typy *výběru turnajem*. Ty probíhají tak, že se náhodně vybere několik jedinců z populace, kteří mezi sebou následně "soupeří"na základě fitness hodnot. Typů turnajových souborů je několik, lišících se například v počtu jedinců účastnících se turnaje, či pravidel, podle kterých vyhrává některý z jedinců.

Binární turnaj (binary tournament) přichází s jednoduchým principem. Jak je z názvu zřejmé, algoritmus vybere vždy dva jedince, ze kterých ten lepší vyhrává a účastní se reprodukce.

Pravděpodobnostní binární turnaj (probabilistic binary tournament) je zobecněním předcházejícího. Zavádí navíc parametr $0.5 < p_{better} < 1$ udávající, s jakou pravděpodobností vyhraje lepší jedinec (v případě prostého binárního turnaje je $p_{better} = 1$)

Porovnání metod

Pokud bychom hledali nějaké studie zaobírající se porovnáním jednotlivých metod pro použití v reálných aplikacích, můžeme zmínit například článek [13]. Jeho cílem bylo porovnání metod škálování fitness funkcí pro použití v optice. Testy byly provedeny nad čtyřmi různými fitness funkcemi použitím těchto metod: pořadová selekce, selekce ořízntutím, lineární a exponenciální škálování. V závěru autoři uvádí doporučení, že pokud je cílem najít rychle řešení, je vhodné použít exponenciální škálování. Na druhou stranu, pokud problém obsahuje více extrémů a je potřeba zachovat populační diverzitu k jejich prozkoumání, měla by se použít selekce ořízntutím.

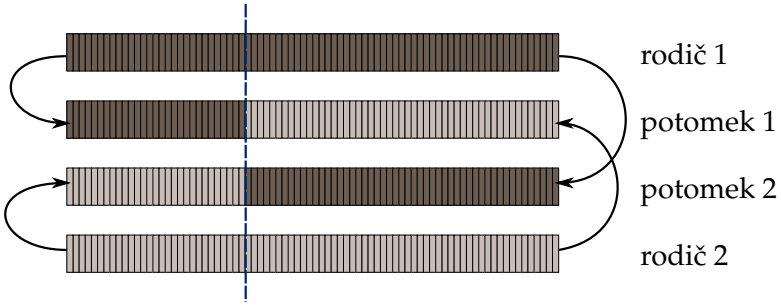
4.1.3 Křížení

Křížení je ovlivněno parametrem pravděpodobnosti křížení p_{xo} určujícím, jaké procento potomků bude vytvořeno tímto procesem. Při $p_{xo} = 100\%$ jsou všichni potomci vytvoření křížením, naproti tomu $p_{xo} = 0\%$ znamená, že žádný potomek nebude takto vytvořen. Zbylá část rodičů vytvoří potomky pomocí svých kopií.

Pro chromozom reprezentovaný binárním řetězcem existuje několik běžně používaných verzí:

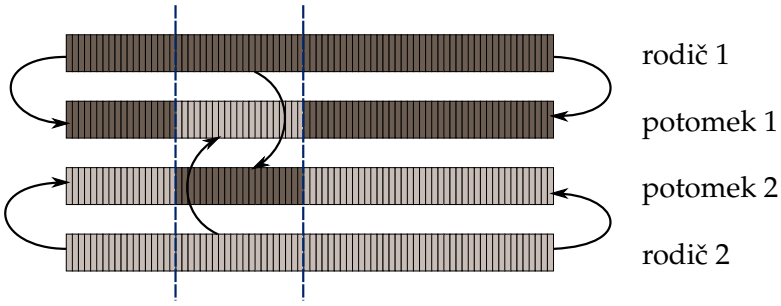
- jednobodové křížení
- dvoubodové křížení
- vícebodové křížení
- uniformní křížení

První varianta, *jednobodové křížení*, definuje pouze jeden náhodný bod mezi oběma jedinci, za kterým dojde u potomků k prohození genetického kódu obou rodičů. Jinak řečeno, první potomek bude do bodu křížení obsahovat bity prvního rodiče, zatímco za tímto bodem bity druhého rodiče. U druhého potomka probíhá křížení opačně, takže začíná bity druhého rodiče, pak prvního.



Obrázek 11: Jednobodové křížení

Další varianta, *dvoubodové křížení*, zavádí takovéto body dva. Rozdíl spočívá v tom, že za druhým bodem se opět prohodí přidělení rodičů k potomkům.

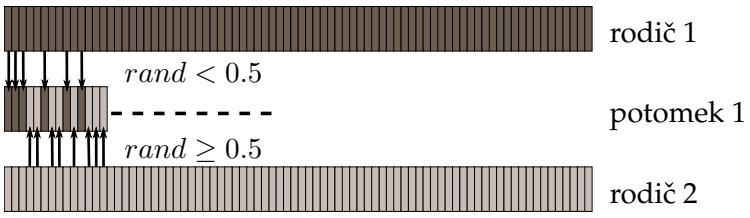


Obrázek 12: Dvoubodové křížení

U *uniformního křížení* je pokaždé vygenerována binární maska o délce chromozomu výsledného potomka. Celý proces pak probíhá tak, že pokud je na masce na pozici x:

- 0, dojde ke zkopírování bitu na pozici x od prvního rodiče,
- 1, dojde ke zkopírování bitu na pozici x od druhého rodiče.

Druhý potomek se vytváří tak, že má bity na každé pozici pocházející od opačného rodiče než jak tomu je u prvního potomka. To se provede nejjednodušeji prohozením pořadí rodičů, nebo také negací celé binární masky.

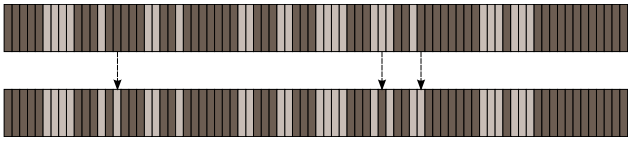


Obrázek 13: Uniformní křížení

4.1.4 Mutace

Mutace je klasicky brána jako méně významný operátor, vnášející malé množství náhodného hledání a snažící se zachovat populační rozmanitost u vysoce konvergované populace. Její vliv se reguluje parametrem *pravděpodobnosti mutace* p_m udávajícím, jak velká část genetického materiálu celé populace bude změněna.

Naproti tomu v [10] je zmíněno několik studií ukazujících, že samotnou mutaci v podání tzv. „naivní evoluce“ nezle brát tak nevýznamně, a že mutace je více významná než se původně myslelo.



Obrázek 14: Mutace chromozomu

4.1.5 Reprezentace chromozomu

Základní teorie genetických algoritmů používá k reprezentaci jedinců binární řetězce. Postupem času ale byly provedeny různé experimenty s reprezentacemi majícími kardinalitu informace uchované v jednom genu větší než 2.

Mezi jeden ze způsobů řešení patří použití reálných čísel. Tato změna posunuje reprezentaci blíže k řešenému problému a hodí se tak například pro problémy typu optimalizace funkce. Jako výhody při použití lze zmínit odpadající nutnost převodu čísel do binární reprezentace s požadovanou přesností, čímž je také délka chromozomu pevná nemění se s přesností, jednodušeji se navrhují procedury ošetřující různá omezení parametrů.

Nový přístup nicméně vyžaduje použití nových metod křížení a mutace. V [1] je uvedena studie kladoucí si za cíl porovnání s klasickou binární reprezentací a představuje také několik nových operátorů křížení a mutace k tomu vyvinutých.

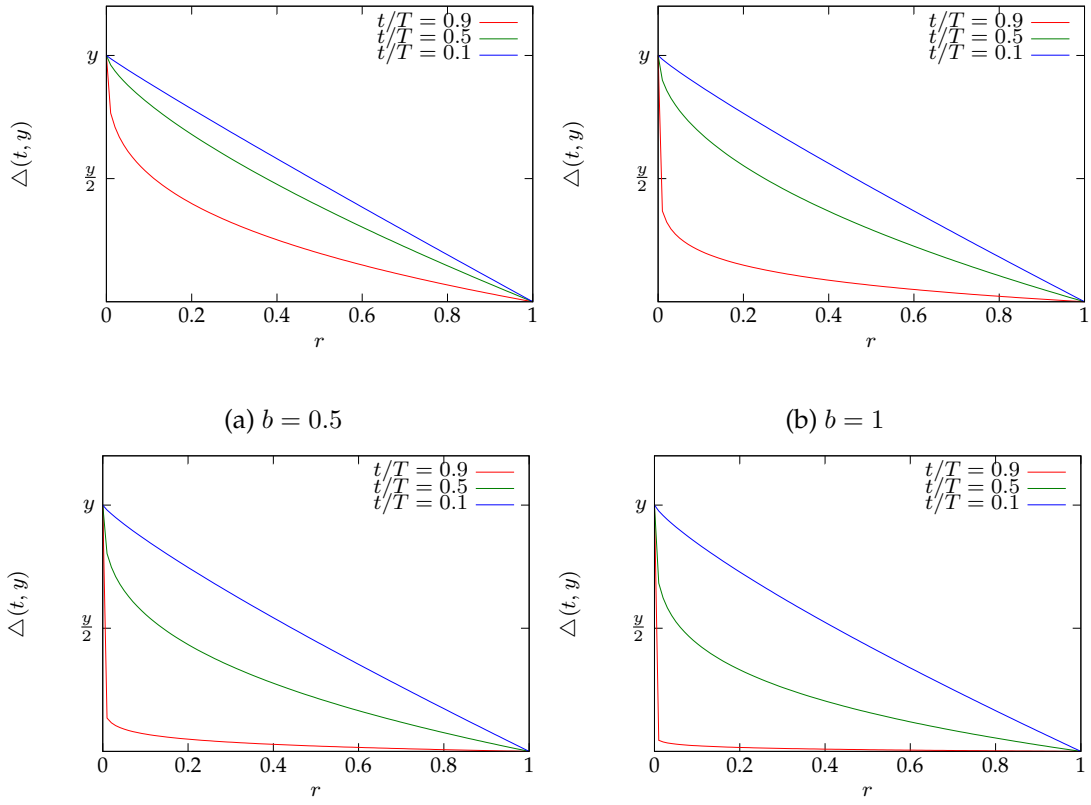
Prvním takovýmto operátorem je *neuniformní mutace*, pracující tak, že gen g_m chromozomu $c = \{g_0, g_1, \ldots, g_m, \ldots, g_n\}$ určený k mutaci bude upraven podle pravidla:

$$g'_m = \begin{cases} g_m + \triangle(t, u_m - g_m) & \text{pokud náhodný bit je 0} \\ g_m + \triangle(t, g_m - l_m) & \text{pokud náhodný bit je 1} \end{cases} \tag{13}$$

kde u_m je horní mez domény atributu g_m
l_m je dolní mez domény atributu g_m
$\triangle(t, y)$ je funkce vracející takovou hodnotu z intervalu $\langle 0; y \rangle$, že pravděpodobnost hodnoty blízké nule roste se stoupajícím t

Jako funkce $\triangle(t, y)$ byla použita:

$$\triangle(t, y) = y * \left(1 - r^{(1-\frac{t}{T})^b}\right) \tag{14}$$



(a) $b = 0.5$ (b) $b = 1$

(c) $b = 1.5$ (d) $b = 2$

Obrázek 15: Rozdělení hodnot funkce $\triangle(t, y)$ podle rovnice 14 pro různé hodnoty parametru b

kde r je náhodné číslo z intervalu $\langle 0; 1 \rangle$
t aktuální generace
T je maximální počet generací
b je parametr určující stupeň nerovnoměrnosti

Druhým operátorem je *aritmetické křížení*, fungující jako lineární kombinace dvou vektorů. Ta pro dva rodiče g_1 a g_2 vytvoří potomky g_1' a g_2' tak, že

$$\begin{aligned} g_1' &= a * g_2 + (1 - a) * g_1 \\ g_2' &= a * g_1 + (1 - a) * g_2 \end{aligned} \tag{15}$$

Nejprve bylo provedeno srovnání za použití operátorů křížení a mutace podobných běžně používaným pro binární reprezentaci. Jediný rozdíl představovala mutace, která namísto změny jednoho bitu vygenerovala nové náhodné číslo z domény argumentu. Při takto navrženém řešení byly výsledky o málo příznivější pro binární řešení, nicméně použití reálných čísel mělo pozitivní vliv na stabilitu díky menší směrodatné odchylce. Rozdíly se začaly ukazovat až s použitím nově vyvinutých operátorů *neuniformní mutace*

a *aritmetického křížení*. Zde nejen že reprezentace reálnými čísly měla lepší stabilitu, ale také se začala projevovat výhodnost tohoto řešení oproti binárnímu.

4.2 Diferenciální evoluce

Diferenciální evoluce je další globální optimalizační technika, tentokrát určená pro práci přímo s celočíselnými vektory. Od svého počátku v roce 1995 došlo k několika změnám které vedly až ke dnešní verzi algoritmu. První verze přitom vznikla při práci K. V. Price na možném použití *genetického žíhání*, pracujícího stejně jako GA s bitovými vektory, na řešení úlohy aproximace Čebyševovými polynomy. Algoritmus nicméně trpěl pomalou konvergencí a správné nastavení parametrů bylo těžké odhadnout.

Následovalo tedy přepsání algoritmu do verze pracující nad vektory reálných čísel, při kterém nejenže byl překonán problém projevující se u genetického žíhání, ale také objeven operátor diferenciální mutace, nad kterým je diferenciální evoluce založena. Autoři také udržují webové stránky[12] s popisem algoritmu, jeho historií, zdrojovými kódy.

Samotný průběh algoritmu je podobný jako například u GA, nicméně s menšími odlišnostmi. Po nastavení hodnot řídících parametrů se inicializuje prvotní populace, poté probíhá samotný proces evoluce v generačních cyklech. První odlišnost spočívá v tom, že v průběhu evoluce je postupně každý jedinec (v tu chvíli nazvaný *aktivní jedinec*) vstupuje do turnaje s tzv. *zkušebním jedincem*. Zde se projevuje druhá odlišnost, protože ten je vytvořen jiným způsobem než potomci u GA. Jednak se při vytváření používá více rodičů než 2, v základní verzi DE jsou to 3, ale také postup prováděných operací je opačný. Mutací je nejprve vytvořen *šumový vektor*, ze kterého se po skřížení s *aktivním jedincem* vytváří už zmíněný *zkušební jedinec*.

Input: $Population_{size}$, $Dimension$, $Generations$, F, CR
Output: S_{best}
$Population$ = CreateInitialPopulation($Population_{size}$, $Dimension$);
Evaluate($Population$);
S_{best} = FindBestSolution($Population$);
for $i = 0$ ***to*** $Generations$ **do**
  foreach $Solution$ ***in*** $Population$ **do**
    $Parents$ = ChooseOtherParents($Population$);
    $Child$ = CreateChild($Solution$, $Parents$, CR, F);
    Evaluate($Child$);
    if *fitness($Child$)* ***is better than*** *fitness($Solution$)* **then**
      $Solution = Child$;
      if *fitness($Child$)* ***is better than*** *fitness(S_{best})* **then**
        $S_{best} = Child$;
      end
    end
  end
end

Algorithm 2: Diferenciální evoluce

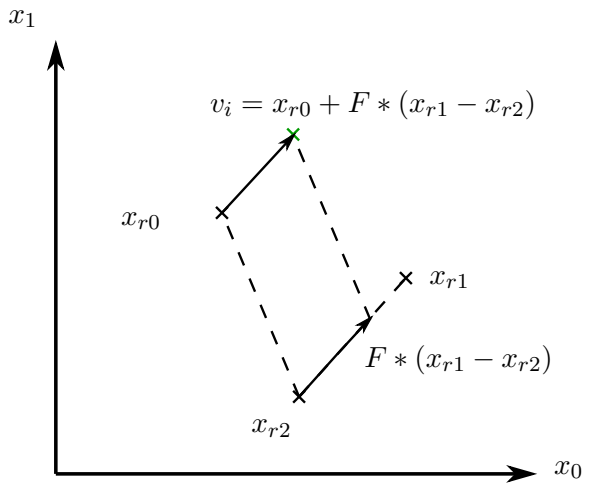
[12] http://www1.icsi.berkeley.edu/~storn/code

4.2.1 Mutace

U základní verze diferenciální evoluce DE/rand/1/bin probíhá mutace tak, že se za pomoci *bázového vektoru* x_{r0} a dvou *rozdílových vektorů* x_{r1} a x_{r2} vytvoří *šumový vektor* v_i

$$v_i = x_{r0} + F * (x_{r1} - x_{r2}) \tag{16}$$

Požadavkem na takto vybrané vektory je, aby byly od sebe rozdílné $i \neq r0 \neq r1 \neq r2 \neq r3$, přičemž nedodržením tohoto pravidla dochází k degradaci kombinace vektorů.



Obrázek 16: Mutace v diferenciální evoluci

Diferenciální evoluce ale není omezena pouze na tento jediný způsob výpočtu šumového vektoru. Existují verze používající odlišnou strategii získání bázového vektoru, různý počet rozdílových vektorů, jiný způsob výpočtu šumového vektoru.

4.2.2 Křížení

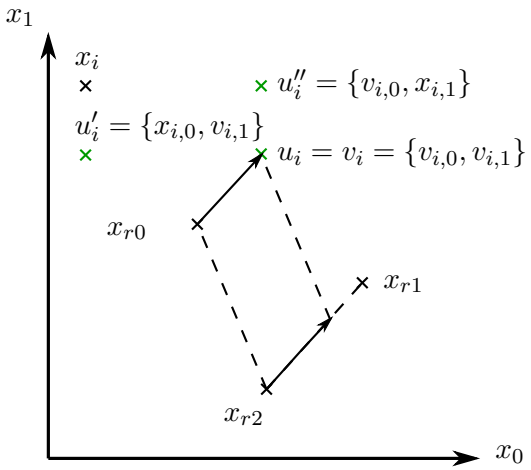
Diferenciální evoluce používá ke křížení variantu *uniformního křížení* tak, jak bylo popsáno u genetických algoritmů. Rozdíl spočívá v ošetření situace, aby výsledný vektor nebyl čistou kopií x_i. Pro každé křížení je proto vybrána jedna náhodná pozice j_{rand} z celého vektoru která zajistí že alespoň jeden parametr se použije z šumového vektoru.

$$j_{rand} = \lfloor r * D \rfloor \tag{17}$$

kde D je dimenze vektoru
r je náhodné číslo z intervalu $\langle 0; 1 \rangle$

Následuje průchod všemi parametry vektorů x_i a v_i a vytvoření *zkušebního vektoru* u_i tak, že se pro všechny pozice j vygeneruje náhodné číslo $r \in \langle 0; 1 \rangle$ a podle porovnání s *prahem křížení* CR se použije parametr z jednoho vektoru

$$u_{i,j} = \begin{cases} v_{i,j} & \text{pokud } r \leq CR \text{ nebo } j = j_{rand} \\ x_{i,j} & \text{jinak} \end{cases} \tag{18}$$



Obrázek 17: Křížení v diferenciální evoluci

4.2.3 Selekce

Selekci u DE lze přirovnat k binárnímu turnaji používanému u genetických algoritmů s tím rozdílem, že jím nejsou vybráni jedinci určení k reprodukci, ale přeživší postupují přímo do další generace. Samotný průběh je takový, že pro každého jedince v populaci x_i, stávajícím se v této chvíli tzv. *aktivním jedincem*, je vytvořen zkušební vektor u_i se kterým vstupuje do turnaje. Výhercem postupujícím do další generace je jedinec s lepší hodnotou fitness funkce.

$$x_{i,g+1} = \begin{cases} u_{i,g} & \text{pokud } f(u_{i,g}) \text{ je lepší než } f(x_{i,g}) \\ x_{i,g} & \text{jinak} \end{cases} \tag{19}$$

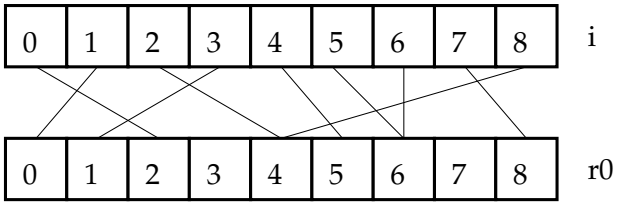
4.2.4 Způsoby výběru bázového vektoru

Náhodný výběr

Klasická varianta **DE/rand/*/*** vybírá index $r0$ náhodně pro každého aktivního jedince na pozici i. Způsoby, jak tohoto dosáhnout, jsou podobné jako u GA. Například výběr pomocí rulety, kde se pro každý výběr vygeneruje náhodné číslo $r \in \langle 0; 1)$ a získá se index $r0$ jako

$$r0 = \lfloor r * PopSize \rfloor \tag{20}$$

Tento způsob nicméně umožňuje, že některý jedinec bude vybrán vícekrát nebo naopak nebude vybrán vůbec. Pomoci si opět můžeme způsobem známým z GA, nicméně mírně upraveným, *SUS*. Rozdíl spočívá v tom, že genetické algoritmy používají sloty úměrně velké fitness funkci, naproti tomu DE nerozlišuje mezi jedinci, takže sloty budou mít stejnou velikost.



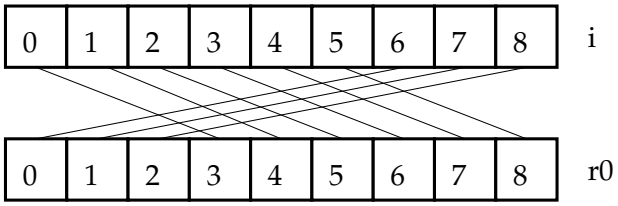
Obrázek 18: Výběr bázového vektoru $r0$ náhodně

V [14] jsou popsány dva způsoby řešení *SUS* u *DE*. Jednodušší varianta *random offset selection* řeší situaci tak, že je nejprve v každé generaci vygenerován náhodný index r_g sloužící jako offset.

$$r_g = \lfloor r * PopSize \rfloor \tag{21}$$

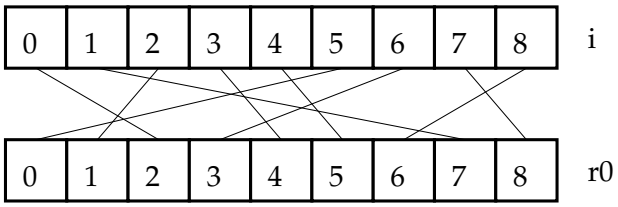
Poté je pro každého aktivního jedince vypočítán index bázového vektoru $r0$ jako posun o hodnotu r_g.

$$r0 = \lfloor (i + r_g) \% PopSize \rfloor \tag{22}$$



Obrázek 19: Výběr bázového vektoru $r0$ posunutím

Druhá varianta *permutation selection* přistupuje k problému jinak. Z pole indexů do populace se vytvoří permutace, která se následně používá jako pole ukazatelů pro výběr. Pro získání indexu $r0$ pro aktivního jedince s indexem i stačí získat hodnotu v poli s permutacemi na pozici i.



Obrázek 20: Výběr bázového vektoru $r0$ permutací

Výběry založené na hodnotě fitness funkce

Mimo základní verze vybírající indexy $r0$ náhodně, byly i zde vyvinuty metody selekce na základě hodnoty fitness tak, jak tomu je například u GA.

Nejjednodušší varianta, **DE/best/*/***, vybírá vždy za bázový vektor jedince s nejlepší fitness hodnotou v současné generaci, takže bude prohledáváno pouze jeho okolí.

$$r0 = best : \forall i \in \{0; PopSize - 1\}; f(x_{best}) \text{ je stejný nebo lepší než } f(x_i) \tag{23}$$

Výběr náhodného nebo nejlepšího jedince jako bázový vektor $r0$ tvoří hranici mezi dvěma extrémy. První zmíněný sice zachovává populační rozmanitost v průběhu evoluce, nicméně je potlačena konvergence. Druhá metoda naopak nedbá na diverzitu jedinců, ale je vyvíjen tlak směrem vždy k nejlepším řešením, s čímž také stoupá problém uvíznutí v lokálním minimu. Postupem času však bylo vyvinuto několik modifikací snažících se obejít problém několika způsoby.

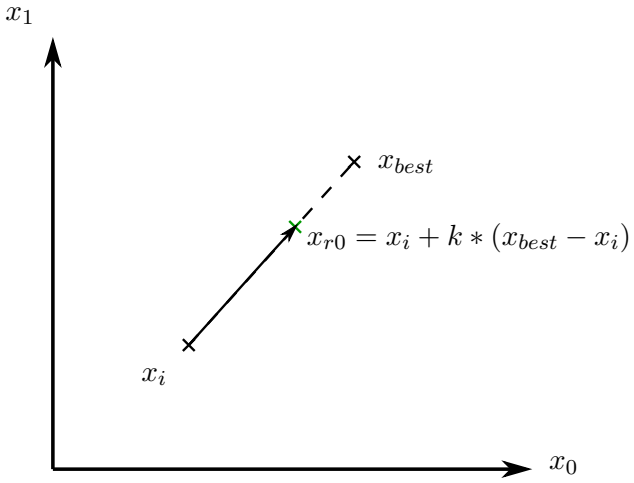
Pokud nechceme vytvářet tak velký selekční tlak jako u *DE/best/*/**, je možnost použít jeho upravenou, **DE/better/*/***. Striktní podmínka na výběr vždy nejlepšího jedince je nahrazena méně přísným výběrem. Na pozici bázového vektoru $r0$ stačí najít takového jedince, který je minimálně stejně dobrý nebo lepší než aktivní x_i.

$$r0 = better : f(x_{better}) \text{ je stejný nebo lepší než } f(x_i) \tag{24}$$

Jiný přístup používá varianta **DE/target-to-best/*/***, která místo toho, aby použila jedince z populace, vytvoří nový vektor na cestě od aktivního jedince k nejlepšímu řešení.

$$x_{r0} = x_i + k * (x_{best} - x_i) \tag{25}$$

Parametr $k \in \langle 0; 1 \rangle$ určuje jak blízko se od aktivního jedince x_i posuneme k nejlepšímu x_{best}. Pro $k = 0$ se bázový vektor x_{r0} bude rovnat aktuálnímu x_i, naproti tomu při $k = 1$ se bude rovnat nejlepšímu x_{best}. Samotný parametr k přitom může být konstanta, proměnná vygenerovaná pro každé x_{r0} nebo i pro každý parametr.



Obrázek 21: Vytvoření bázového vektoru x_{r0} metodou *DE/target-to-best/*/**

Variantu podobnou turnajovému výběru jedinců popsanou u genetických algoritmů je **DE/BoR/*/*** [15]. Samotného turnaje se účastní stejný počet jedinců, jaký je potřebný k vytvoření šumového vektoru. Vítěz souboje je zvolen bázovým vektorem $r0$, zbylí jedinci se stanou rozdílovými vektory.

Ve studii [15] bylo provedeno srovnání s oběma základními verzemi DE na nízko i vysoce dimenzionálních funkcích. Výsledky ukázaly, že v unimodálních nízko dimenzionálních funkcích se lépe chová *DE/best/1/**, naopak u multimodálních funkcí se lépe chovají verze *DE/rand/1** a *DE/BoR/1/**. Pro funkce s vysokou dimenzí se ještě více projevil rozdíl mezi algoritmy ve prospěch *DE/BoR/1**.

Jeden z novějších přístupů [16], v původním textu nazvaný **tradeoff strategy**, byl prezentován při vývoji nové kalibrační metody pro robota. Obě základní verze DE předčil nejen v rychlosti konvergence, ale také v kvalitě získaných výsledků.

$$v_i = \frac{x_{r1} + x_{best}}{2} + F * (x_{r2} - x_{r3}) \tag{26}$$

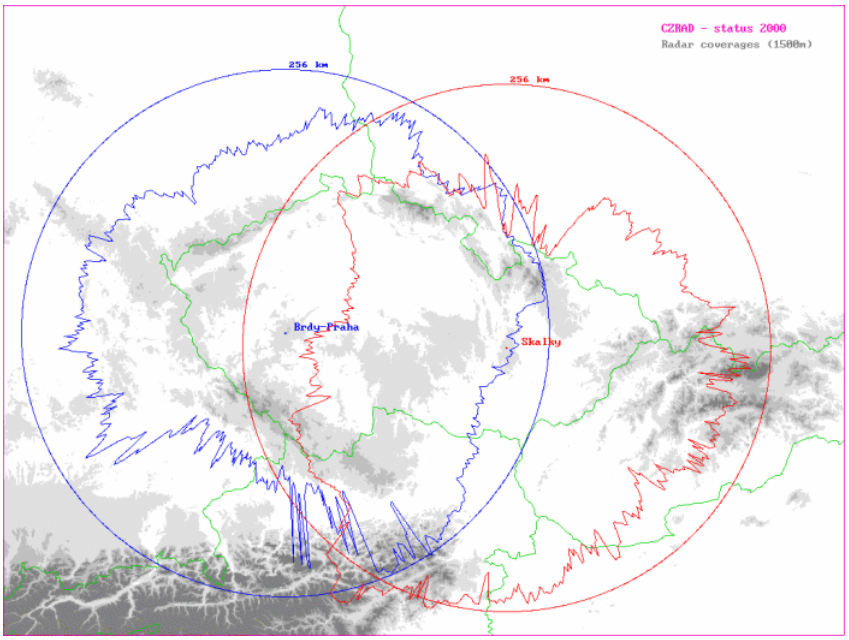
Pokud se podíváme na rovnici zjistíme, že se podobá variantě *DE/target-to-best/*/** s tím rozdílem, že bázový vektor se nevytváří na cestě mezi nejlepším x_{best} a aktivním x_i řešením, ale

$$x_{r0} = \frac{x_{r1} + x_{best}}{2} \tag{27}$$

5 Naše metoda

5.1 Data poskytovaná ČHMÚ

Radarová data[13] poskytovaná ČHMÚ jsou kombinací dat získávaných z radarové sítě CZRAD [17]. V současné době jsou v provozu dvě stanice umístěné na opačných koncích republiky ve vzdálenosti přibližně 215 km od sebe, přitom každá z nich je schopná mapovat své okolí až do vzdálenosti 250 km.

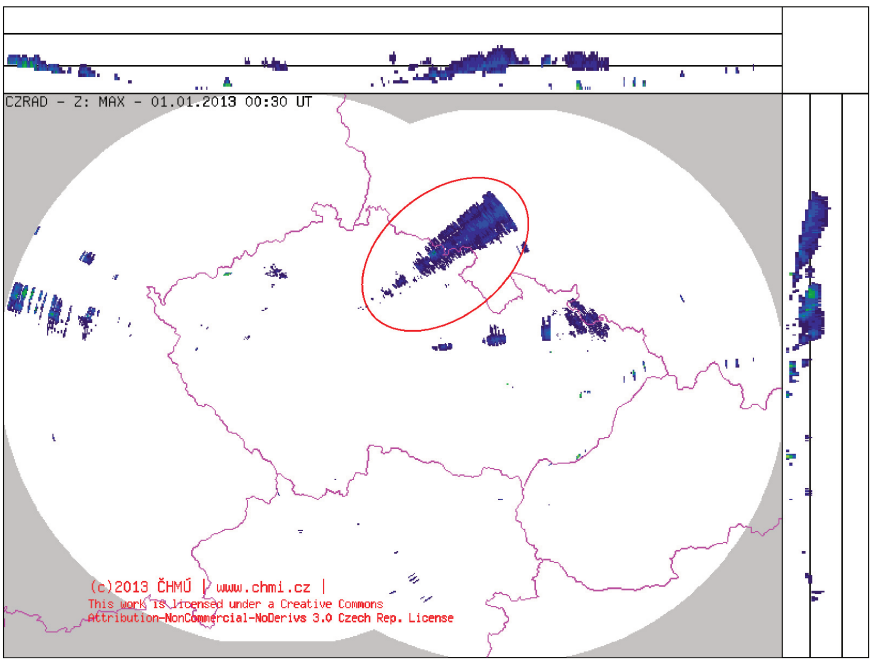


Obrázek 22: Maximální dosahy meteorologických radarů ČHMÚ a dosahy pro určování intenzit srážek (do výšky 1500 m nad terénem). Zdroj: [17]

Tyto radary jsou schopny zjišťovat aktuální srážkovou situaci tím, že do svého okolí vysílají elektromagnetické pulzy ve tvaru kužele o délce trvání v řádu µs s frekvencí opakování ve stovkách Hz. Anténa následně přijímá zpětné odrazy od objektů, které mohou být dvojího typu. Jedná se jednak o meteorologické objekty, do kterých patří srážky, ale také jiné objekty, jako je například terén. Informace získané při příjmu signálu slouží k identifikaci cíle. Z horizontální a vertikální polohy antény při příjmu zjistíme polohu cíle, odstup mezi vysláním pulzu a samotným příjmem udává vzdálenost, síla signálu určuje radarovou odrazivost. Odrazivost slouží následně k výpočtu průměru vodních kapek.

Tento přístup má ale také svou stinnou stránku způsobenou technickými parametry. Meteorologické radary totiž pracují v pásmu 5 GHz (v našem případě 5645 MHz a 5630 MHz) ve kterém také fungují zařízení podle specifikace 802.11a. Tato situace, obrázek 23, má za následek výskyt méně či více rušivých elementů v dodávaných datech.

[13] http://portal.chmi.cz/files/portal/docs/meteo/rad/data_jsradview.html



Obrázek 23: Rušení způsobené zařízeními pracující ve stejném frekvenčním pásmu jako meteorologické radary

5.2 Popis

Naši metodu jsme se rozhodli založit čistě na radarových snímcích poskytovaných ČHMÚ. Abychom z těchto dat byli schopni předpovídat budoucí vývoj srážkové situace, použili jsme jednoduchý nápad. Pro předpověď následujících hodnot srážek stačí znát několik předešlých hodnot na daném místě. Za účelem poskytnutí také nějakého širšího kontextu o situaci probíhající v daném bodě jsme zahrnuli navíc jeho přímé okolí.

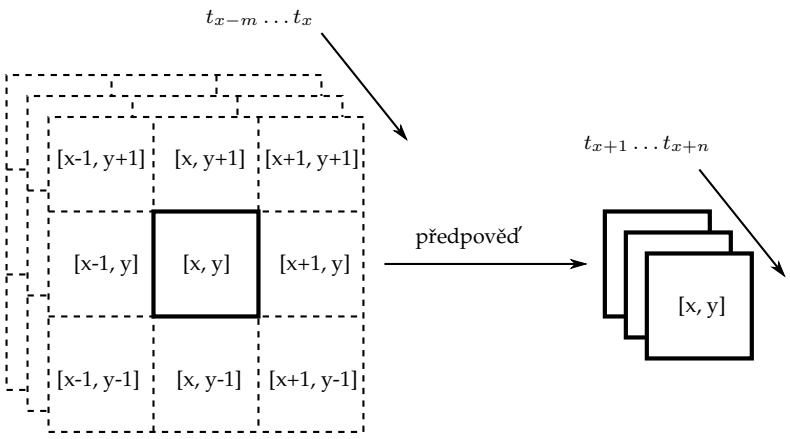
Celkem bylo vyzkoušeno několik variant tohoto přístupu, přičemž všechny používaly vrstvenou neuronovou síť. Lišily se jednak použitou aktivační funkcí neuronové sítě, ale také způsobem specifikace kontextové informace. Ta byla v základní variantě, obrázek 24a, předávána stejně jako pro hodnotu bodu na kterém se prováděla předpověď, tedy surovou hodnotou radiolokační odrazivosti. Druhá varianta byla mírně modifikovaná. Úprava spočívala ve změně reprezentace okolních bodů, obrázek 24b. Hodnotě středového bodu sice zůstala absolutní hodnota, ale okolní hodnoty byly reprezentovány jako relativní změna oproti předchozímu snímku.

Úroveň srážek na daném místě je v poskytovaných radarových datech reprezentována hodnotou radiolokační odrazivosti ve stupnici od $4\,\mathrm{dBZ}$ do $60\,\mathrm{dBZ}$ s velikostí kroku $4\,\mathrm{dBZ}$ pomocí 15 různých barevných odstínů. Z těchto hodnot lze pomocí logaritmické stupnice vyjádřit také hodnotu aktuálních srážek na daném místě v $\mathrm{mm\,h^{-1}}$. Protože v předpovědích pomocí neuronové síti byly použity dva typy aktivačních funkcí, hyperbolický tangens a sigmoida, bylo potřeba převést hodnoty odrazivosti do intervalu vhodného pro zpracování. V našem případě tedy $\langle 0; 1\rangle$.

(a) Bod i okolí definováno absolutními hodnotami

(b) Okolí definováno jako relativní změna oproti předchozímu snímku

Obrázek 24: Způsoby předání informace o situaci v předešlých snímcích



Obrázek 25: Vstupní a výstupní data pro neuronovou síť

5.3 Experimenty

5.3.1 Ověření navržených modelů

Abychom nějakým způsobem ověřili základní funkčnost takto navrženého modelu, vyvinuli jsme pro účely testování jednoduchý generátor mraků. Na něm bylo následně provedeno srovnání tří navrhovaných typů modelů popsaných v tabulce 2.

bod pro předpověď	okolní body	aktivační funkce
absolutní hodnota	absolutní hodnota	sigmoida
absolutní hodnota	absolutní hodnota	hyperbolický tangens
absolutní hodnota	relativní hodnota	hyperbolický tangens

Tabulka 2: Typy navrhovaných modelů

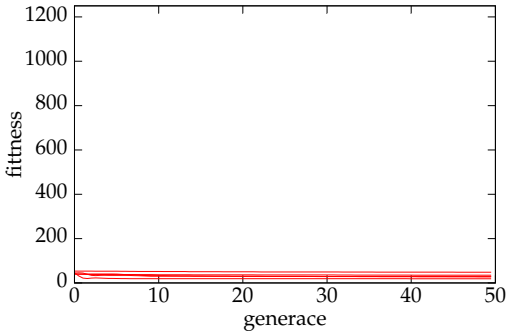
Jak lze vidět na obrázku 26a, první zkoušený model nedopadl vůbec dobře. Po prvních pár generacích se fitness hodnota usadila na hodnotách vyšších než u ostatních dvou modelů a evoluce stagnovala. Jako rušivý faktor zde působila aktivační funkce. Pouhá změna ze sigmoidy na hyperbolický tangens podávala velice dobré výsledky. Podobných výsledků bylo dosaženo i s třetím modelem. Ten používá také hyperbolický tangens jako aktivační funkci, ale okolní body jsou relativní změna oproti předchozímu snímku.

Pokud srovnáme poslední dva zmíněné modely vzhledem k hodnotám *Precision* (obrázek 53), *Recall* (obrázek 54) a *F-score* (obrázek 27), popsaných v kapitole 2.2, zjistíme, že podávají podobné výsledky. Třetí model ale znatelně lepší v hodnotách *Precision* i při nižší fitness. Díky tomu bylo celkové *F-score* hlavně pro čas $t+1$ o něco lepší než u druhého modelu. Pro další použití byl tedy vybrán poslední model.

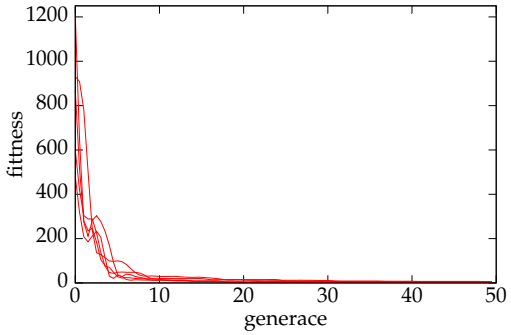
Během těchto testů bylo také vyzkoušeno několik různých konfigurací neuronové sítě. Podle dosažených výsledků byla použita konfigurace 18-45-30-2, tedy mající osmnáct vstupů (bod a jeho okolí pro dva časové intervaly), dvě skryté vrstvy a dva výstupy (předpověď pro daný bod v následujících dvou časových intervalech).

5.3.2 Algoritmy učení

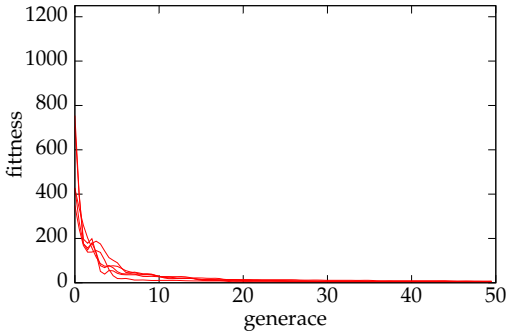
K naučení neuronové sítě bylo vybráno několik dobře známých evolučních algoritmů, u nichž bylo potřeba zjistit, jak dobře či špatně si poradí s řešením námi zadaného problému. Úkolem bylo nalezení vhodného nastavení vah neuronové sítě s pevně danou strukturou. Pro základní přehled o jejich schopnosti nalézt optimální řešení, ale také z časových důvodů, byly provedeny nad daty generovanými testovacím generátorem. Dále by také bylo vhodné vyhnout se náhodným výkyvům v evoluci, přece jenom nastavení vah neuronové sítě probíhá náhodně a algoritmy nepatří k deterministickým, proto je provedeno vždy 5 běhů algoritmu.



(a) Okolní body jako absolutní hodnoty, sigmoida

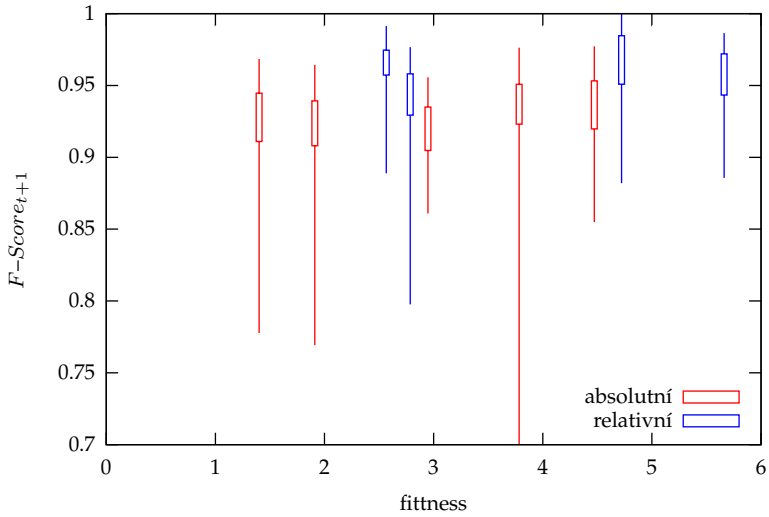


(b) Okolní body jako absolutní hodnoty, hyperbolický tangens

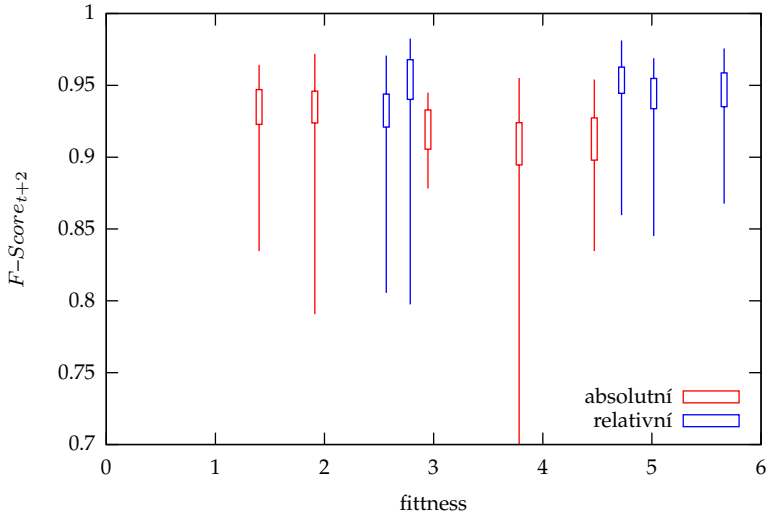


(c) Okolní body jako relativní hodnoty, hyperbolický tangens

Obrázek 26: Porovnání učení sítě při použití různých modelů

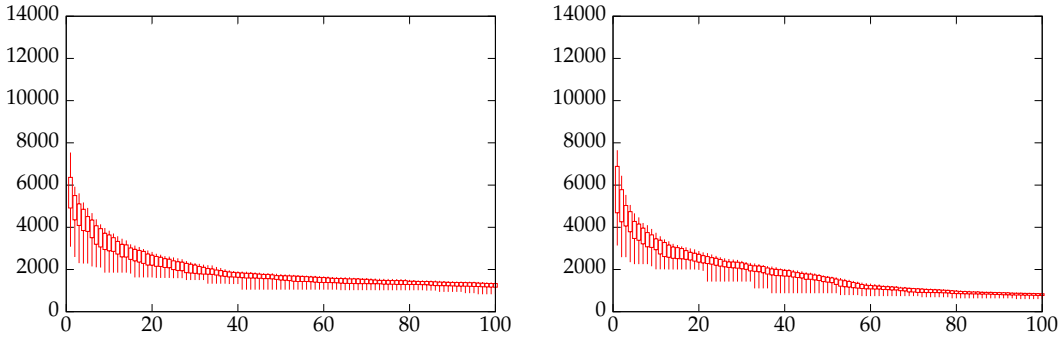


(a) $F\text{–}Score_{t+1}$



(b) $F\text{–}Score_{t+2}$

Obrázek 27: $F\text{–}Score$ pro model s absolutními a relativními hodnotami okolních bodů pro časy $t+1$ a $t+2$



(a) výchozí ($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$, RWS)

(b) upravené ($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.005$, SUS)

Obrázek 28: Porovnání průběhů evoluce genetickým algoritmem s výchozími a upravenými parametry při binární reprezentaci jedinců

Genetický algoritmus

Genetický algoritmus s binární reprezentací jedinců je ze zde zmíněných algoritmů nejstarší. Bohužel převod řešení do podoby binárního řetězce s sebou nese jednak řešení přesnosti použité při převodu jednotlivých čísel do binární podoby, ale se vzrůstající přesností následně narůstá délka chromozomu. To jednak prodlužuje délku trvání evolučních cyklů, ale také zvětšuje množinu prohledávaných řešení.

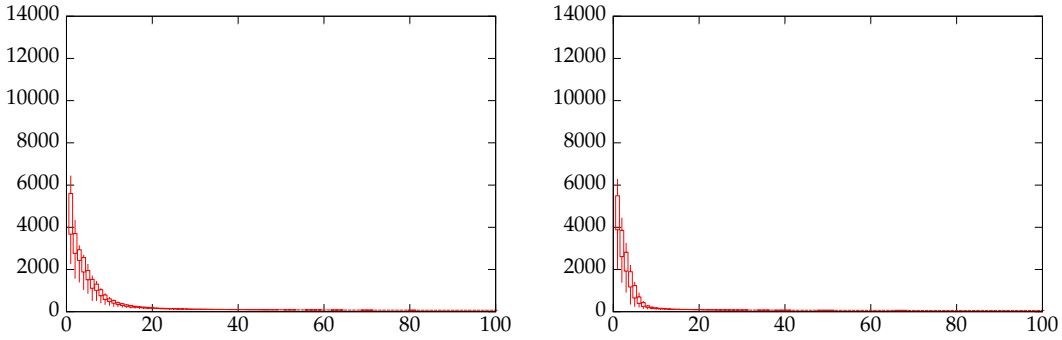
parametr	rozsah hodnot	výchozí hodnota
$PopSize$	≥ 10	100
p_m	$\langle 0; 1 \rangle$	0.01
p_{xo}	$\langle 0; 1 \rangle$	0.95
a	$\langle 0; 1 \rangle$	0.10
b	≥ 0	2.00

Tabulka 3: Genetický algoritmus, výchozí nastavení

Jak lze vidět na obrázku 28, nastavení jednotlivých parametrů se nijak znatelně neprojevilo na průběhu evoluce. Změna nastala až použitím reprezentace jedinců používající čísla s plovoucí desetinnou čárkou, obrázek 29. Jak se později ukáže, tato metoda předčila ostatní v rychlosti s jakou byla schopna nalézat optimální řešení se stejnou kvalitou.

Diferenciální evoluce

Jak již bylo zmíněno dříve, diferenciální evoluce v základní verzi *DE/rand/1/bin* používá náhodný výběr bázového vektoru $r0$ a její nastavení lze ovlivnit parametry uvedenými



(a) výchozí ($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$, $a = 0.1$, $b = 2.0$, SUS)

(b) upravené ($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$, $a = 0.2$, $b = 2.0$, SUS)

Obrázek 29: Porovnání průběhů evoluce genetickým algoritmem s výchozími a upravenými parametry při reprezentaci jedinců reálnými čísly

v tabulce 4. Ve výchozím nastavení bohužel algoritmus konvergoval velice pomalu, spíše vůbec. Nezbývalo nic jiného než ověřit, zda se tento problém nedá odstranit.

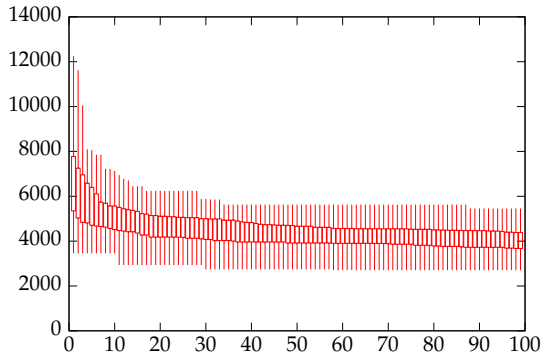
parametr	rozsah hodnot	výchozí hodnota
$PopSize$		100
CR	$\langle 0; 1 \rangle$	0.80
F	$\langle 0; 2 \rangle$	0.40

Tabulka 4: Diferenciální evoluce, výchozí nastavení

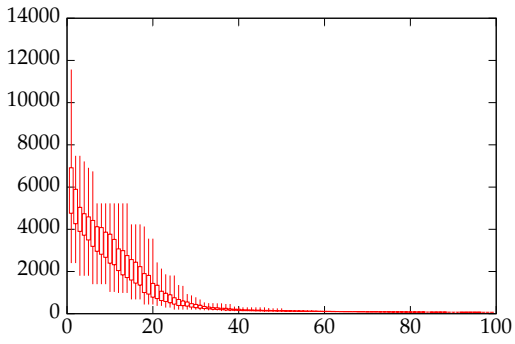
První test byl zaměřen na nastavení parametru pro práh křížení CR. Podle [4] je pro separabilní problémy doporučeno nastavit $CR \ll 1$, naproti tomu pro neseparabilní problémy $CR \approx 1$, přičemž doporučené nastavení je v intervalu $0.8 - 0.9$. Výsledky toto potvrzovaly. Stejný test byl proveden pro mutační konstantu, u které [4] zmiňuje pouze doporučené nastavení v rozsahu $0.3 - 0.9$.

Kromě základní verze diferenciální evoluce *DE/rand/1/bin* byly vyzkoušeny také dvě varianty používající odlišný přístup k výběru bázového vektoru. Oba dva, *DE/BoR/1/bin* a *tradeoff strategy*, ve studiích je prezentující [15, 16] prokazovaly lepší výsledky oproti základní verzi, *DE/BoR/1/bin* hlavně pro funkce s vysokou dimenzí.

Výsledky lze vidět na obrázku 38 a pro verzi *DE/BoR/1/bin* potvrdily slibované vlastnosti. Algoritmus si v průběhu evoluce zachoval diverzitu populace, ale také se urychlila jeho konvergence. Druhý prezentovaný algoritmus tak úspěšný nebyl. Zachoval si sice rozmanitost populace, konvergence byla i rychlejší než v případě *DE/BoR/1/bin*, nicméně v nejlepším případě nacházel pouze řešení s dvojnásobnou hodnotou fitness oproti *DE/-BoR/1/bin*.



(a) výchozí ($DE/rand/1/bin$, $PopSize = 100$, $CR = 0.8$, $F = 0.4$)



(b) upravené ($DE/BoR/1/bin$, $PopSize = 100$, $CR = 0.95$, $F = 0.2$)

Obrázek 30: Porovnání průběhů evoluce pomocí algoritmu *DE* s výchozími a upravenými parametry

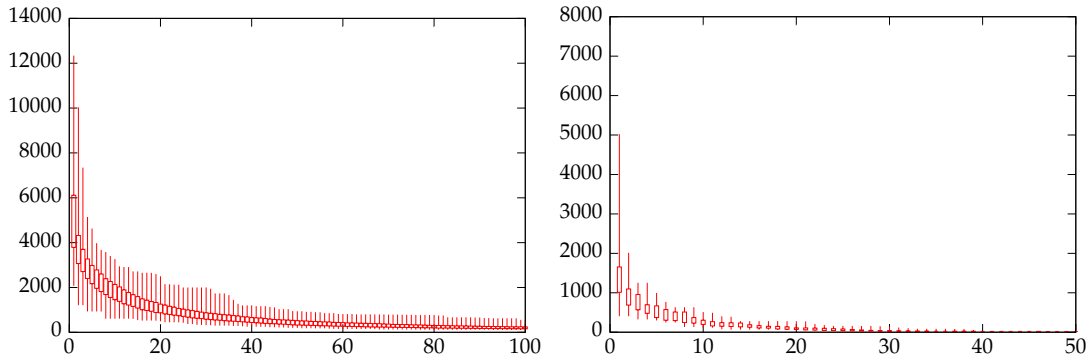
5.3.3 SOMA

Z výsledku spuštění algoritmu ve výchozím nastavení bylo zřejmé, že podává o mnoho lepší výsledky než diferenciální evoluce a genetický algoritmus v binární verzi. Dalo se tedy očekávat, že evoluce se bude dát ovlivnit k lepším výsledkům.

Nastavení parametru PTR bylo přitom docela překvapením. Podle [4] se optimální hodnota pohybuje okolo 0.1, přičemž se vzrůstající hodnotou má stoupat konvergence algoritmu k lokálním extrémům. V případech problémů s nízkou dimenzí a vysokým počtem jedinců je možné nastavit tuto hodnotu na $0.7 - 1.0$. Po provedení testů, výsledek lze vidět na obrázku 40, se právě vyšší hodnota projevovala pozitivně vzhledem ke kvalitě získaných řešení. Přitom daný problém není vůbec nízkodimenzionální, ani počet jedinců nebyl vysoký.

parametr	rozsah hodnot	výchozí hodnota
$PopSize$	≥ 10	100
PTR	$\langle 0; 1 \rangle$	0.10
$Step$	> 0	0.11
$PathLength$	≥ 1.1	1.10

Tabulka 5: SOMA, výchozí nastavení



(a) výchozí ($PopSize = 100$, $PTR = 0.1$, $Step = 0.11$, $PathLength = 1.1$)

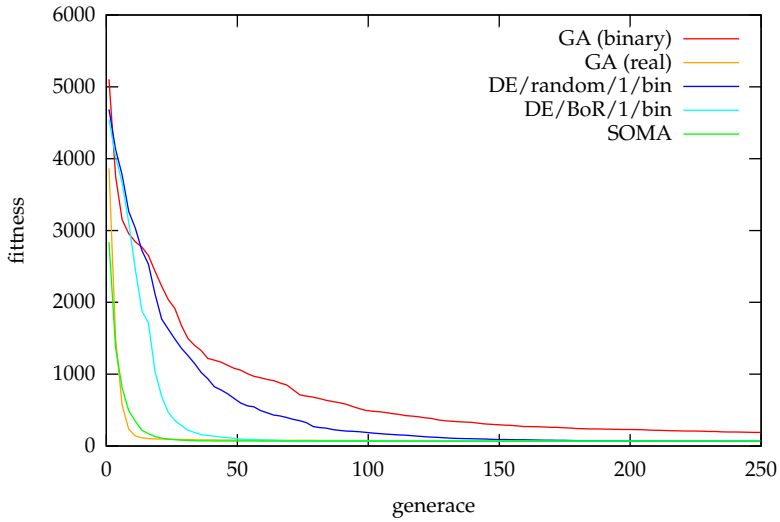
(b) upravené ($PopSize = 20$, $PTR = 0.5$, $Step = 0.112$, $PathLength = 1.25$)

Obrázek 31: Porovnání průběhu evoluce algoritmem *SOMA* s výchozími a upravenými parametry

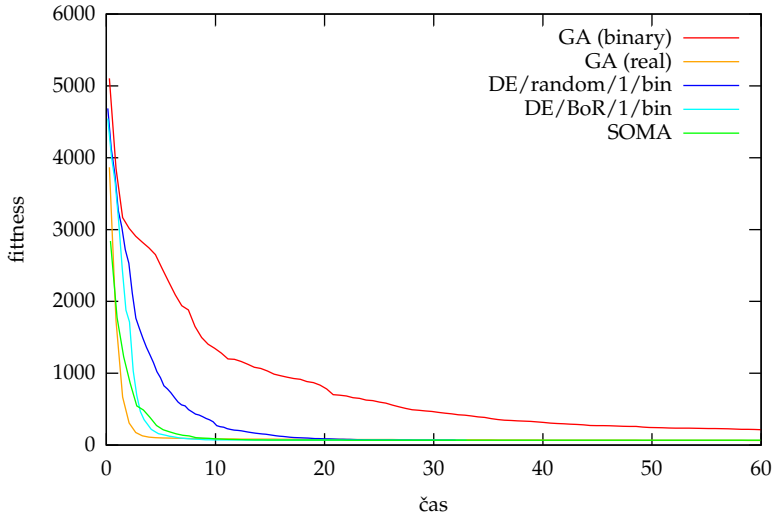
Porovnání metod

Z těchto metod bylo následovně potřeba vybrat jednu, která se bude nejlépe hodit k dalšímu použití nad reálnými daty. Porovnání průběhu evoluce bylo jednak provedeno vzhledem ke generacím, ale také času. Na výsledcích, obrázky 32 a 33, lze vidět, že genetický algoritmus pracující s binární reprezentací jedinců konvergoval nejpomaleji, při použití čísel s plovoucí desetinnou čárkou naopak nejrychleji z ostatních.

SOMA se při porovnání vzhledem ke generacím jevila přibližně stejně jako druhá varianta genetického algoritmu, ale při porovnání vzhledem k času se projevila jedna její vlastnost. Ta spočívá v tom, jakým způsobem funguje její evoluce. Na rozdíl od ostatních variant je totiž v průběhu jedné generace, zde nazvané migrační cyklus, provedeno více ohodnocení nových řešení než v ostatních metodách. V tomto srovnání se tedy jevila přibližně stejně jako diferenciální evoluce ve variantě *DE/BoR/1/bin*.



Obrázek 32: Průběh evoluce implementovaných metod v závislosti na generaci



Obrázek 33: Průběh evoluce implementovaných metod v závislosti na čase

5.3.4 Ověření modelu na reálných datech

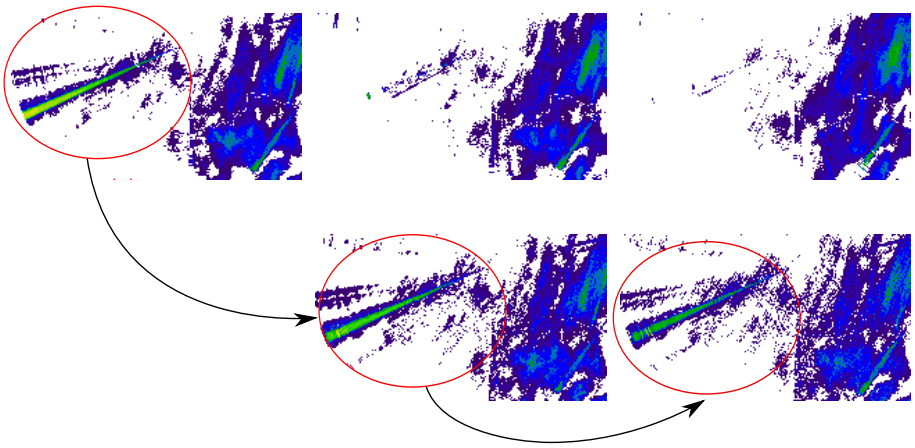
Na reálných datech bohužel predikční schopnost neuronové sítě klesla. Podíl na tomto mělo jednak rušení ve zdrojových datech způsobené zařízeními pracujícími ve stejném frekvenčním pásmu. Rušivé segmenty jimi způsobené se vyskytují nejčastěji v jednom až dvou po sobě následujících snímcích. Protože neuronová síť nemá možnost jak zjistit, že hodnoty na vstupech nejsou validní pro vytváření předpovědi, pracuje s nimi a provede předpověď, jako by se jednalo o normální srážková data. Následkem je, obrázek 34, že v předpovědích zůstane rušení zachováno, i když na skutečných snímcích se už nevyskytuje.

Další překážkou se stal časový rozestup snímků. Posun srážkové události mezi nimi byl větší než námi navržený model dokázal zachytit, takže v základním rozlišení se předpověď jevila jako ustálená na místě posledního dodaného snímku. Řešení se dá nalézt ve vhodném zmenšení dat ve fázi předzpracování. Nicméně to s sebou nese jednak nevýhodu ztráty rozlišení výsledné predikce, ale také pohyb události není ve všech místech konstantní. Ve výsledku ale může nastat situace, že na některých místech bychom tímto přišli o informaci o posunu.

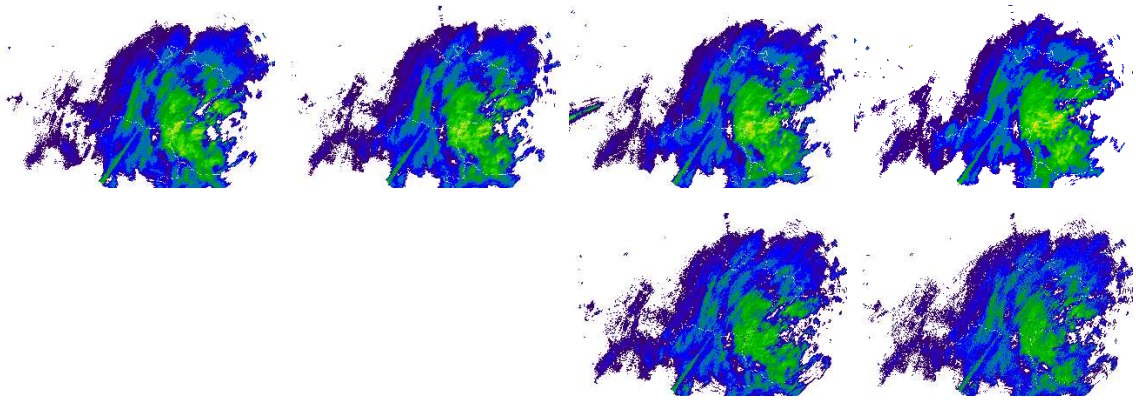
Řešením by také bylo zvětšit poskytované okolí bodu z pouze přímého i na body ve vzdálenosti 2, případně i větší. Negativní dopad tohoto přístupu ale je, že neuronová síť bude muset být také zvětšena, což ve výsledku prodlouží čas potřebný k jejímu ohodnocení při procesu učení.

	Q_1	Avg	Q_3
$Precision_{t+1}$	0.40	0.6	0.93
$Precision_{t+2}$	0.34	0.63	0.91
$Recall_{t+1}$	0.67	0.77	0.96
$Recall_{t+2}$	0.57	0.7	0.94
$F–Score_{t+1}$	0.60	0.76	0.95
$F–Score_{t+2}$	0.54	0.72	0.94

Tabulka 6: Predikční schopnost na reálných datech



Obrázek 34: Vliv rušení ve zdrojových datech na předpovědi
(horní sada snímků jsou zdrojová data, dolní je předpověď z prvního zdrojového snímku na následující dva)



Obrázek 35: Předpověď na reálných datech. Horní sada snímků je průběh zachycený radary, spodní sada je predikce založená na prvních dvou snímcích.

6 Závěr

Cílem práce bylo navrhnout způsob krátkodobé predikce srážek pomocí neuronových sítí pouze s použitím radarových snímků. Jako programovací jazyk použitý k implementaci byl vybrán C#. Během vývoje byly následně implementovány celkem tři algoritmy učení (genetický algoritmus, diferenciální evoluce, SOMA), jeden typ neuronové sítě (dopředná neuronová síť) a tři modely pro predikci.

Pro účely ověření základní funkčnosti námi navržených modelů byl navíc vyvinut testovací generátor mraků, na kterém jsem dále ověřoval také úspěšnost algoritmů učení i závislost dosažených výsledků vzhledem k použité struktuře neuronové sítě.

Z výsledků testů jsem pro další použití nad reálnými daty zvolil dopřednou neuronovou síť v konfiguraci 18-45-30-2, tedy mající osmnáct vstupů (bod a jeho okolí pro dva časové intervaly), dvě skryté vrstvy a dva výstupy (předpověď pro daný bod v následujících dvou časových intervalech). Pro naučení této sítě jsem vybral genetický algoritmus, který místo binárních řetězců používá k reprezentaci genů čísla s plovoucí desetinnou čárkou.

Tato konfigurace, jak již bylo zmíněno, byla schopna produkovat předpovědi, jejichž hodnota $F\text{–}Score$ pro čas t_{x+1} se pohybovala v rozmezí od 0.9 až ≈ 1. Po ověření na reálných datech, kapitola 5.3.4, predikční schopnost bohužel klesla k hodnotám s průměrnou hodnotou ≈ 0.6.

Pro vylepšení stávající metody, na které už během práce nezbyl čas, by bylo vhodné vyzkoušet také rekurentní typ neuronové sítě. Mimo to by se model také mohl upravit tak, aby byl schopen se srovnat s rychlostí pohybu mraků v radarových snímcích. Toho by bylo možné dosáhnout například vhodným předzpracováním radarových snímků, jako jejich rozdělením na regiony se stejnou rychlostí pohybu, či rozšířením kontextové informace každého bodu.

Další překážka, kterou by bylo vhodné odstranit jednak pro získání přesnějšího učení, ale i lepších koncových výsledků, jsou rušení v radarových datech způsobené zařízeními pracujícími ve stejném frekvenčním pásmu.

7 Reference

[1] MICHALEWICZ, Zbigniew. *Genetic algorithms+data structures=evolution programs.* 3rd rev. extended ed. Berlin: Springer, 1996, viii, ix, 387 s. ISBN 35-406-0676-9.

[2] HEATON, Jeff. *Introduction to neural networks for C#.* 2nd ed. St. Louis: Heaton Research Inc, 2008, viii, ix, 387 s. ISBN 16-043-9009-3.

[3] BROWNLEE, Jason. *Clever algorithms: nature-inspired programming recipes.* S.l.: Lulu Com. ISBN 978-144-6785-065.

[4] *Evoluční výpočetní techniky: principy a aplikace.* 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.

[5] HUNG, N. Q., M. S. BABEL, S. WEESAKUL, a N. K. TRIPATHI An artificial neural network model for rainfall forecasting in Bangkok, Thailand. In: *Hydrology and Earth System Sciences* [online]. Volume 13, 2009, s. 1413-1425. DOI: 10.5194.

[6] TOTH, E., A. BRATH a A. MONTANARI Comparison of short-term rainfall prediction models for real-time flood forecasting. In: *Journal of Hydrology*. vol. 239: Elseiver, 2010, s. 132-147.

[7] WANG, Peng, Alan SMEATON, Songyang LAO, Edel O'CONNOR, Yunxiang LING a Noel O'CONNOR Short-Term Rainfall Nowcasting: Using Rainfall Radar Imaging.

[8] HOLLAND, John H. Genetic algorithms. Scientific american, 1992, 267.1: 66-72.

[9] BEASLEY, David, David R. BULL a Ralph R. MARTIN. An Overview of Genetic Algorithms: Part 1, Fundamentals. In: *University Computing*.

[10] BEASLEY, David, David R. BULL a Ralph R. MARTIN. An Overview of Genetic Algorithms: Part 2, Research Topics. In: *University Computing*.

[11] MAN, K.F., K.S. TANG a S. KWONG. *Genetic algorithms: concepts and applications [in engineering design].* In: *IEEE transactions on industrial electronics and control instrumentation*. Volume.43, no.5., Oct 1996, s. 519-534. ISSN 0278-0046. DOI: 10.1109/41.538609.

[12] HOPGOOD, Adrian A. *Intelligent systems for engineers and scientists*. 2nd ed. Boca Raton, Fla.: CRC Press, c2001, 467 p. ISBN 978-084-9304-569.

[13] A. SADJADI, Farzad. *Comparison of fitness scaling functions in genetic algorithms with applications to optical processing*.

[14] PRICE, Kenneth V, Rainer M STORN a Jouni A LAMPINEN. *Differentional evolution: a practical approach to global optimization*. Berlin: Springer, 2005, xix, 538 s. ISBN 35-402-0950-6.

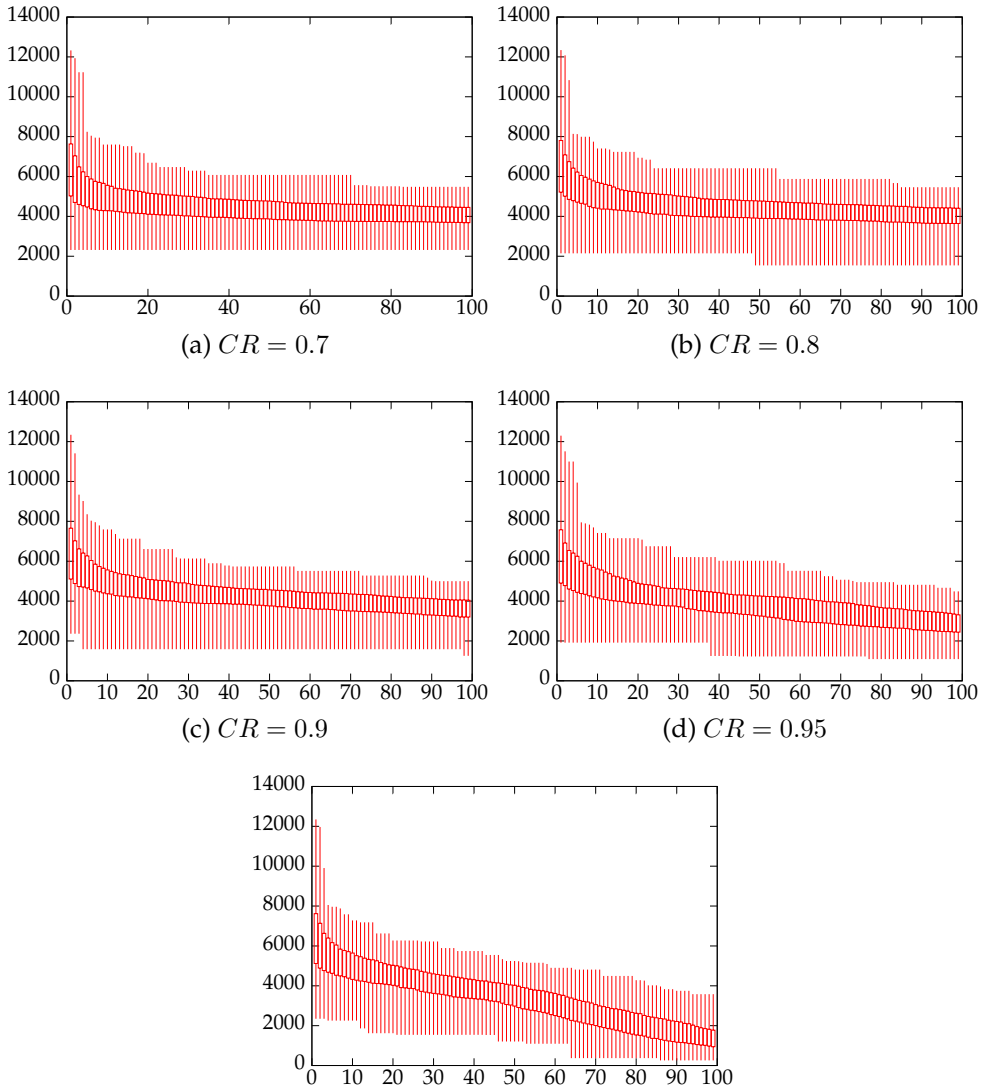
[15] LIN, Chuan, Anyong QING a Quanyuan FENG. *A new differential mutation base generator for differential evolution*. DOI: 10.1007/s10898-010-9535-7.

[16] LIU, Yu, Feng-lei NI, Hong LIU a Wen-fu XU. *Enhancing pose accuracy of space robot by improved differential evolution*. DOI: 10.1007/s11771-012-1095-1.

[17] Radarová síť CZRAD. [online]. [cit. 2013-04-28]. Dostupné z: http://www.chmi.cz/files/portal/docs/meteo/rad/info_czrad/index.html

A Testování parametrů algoritmů

A.1 Diferenciální evoluce

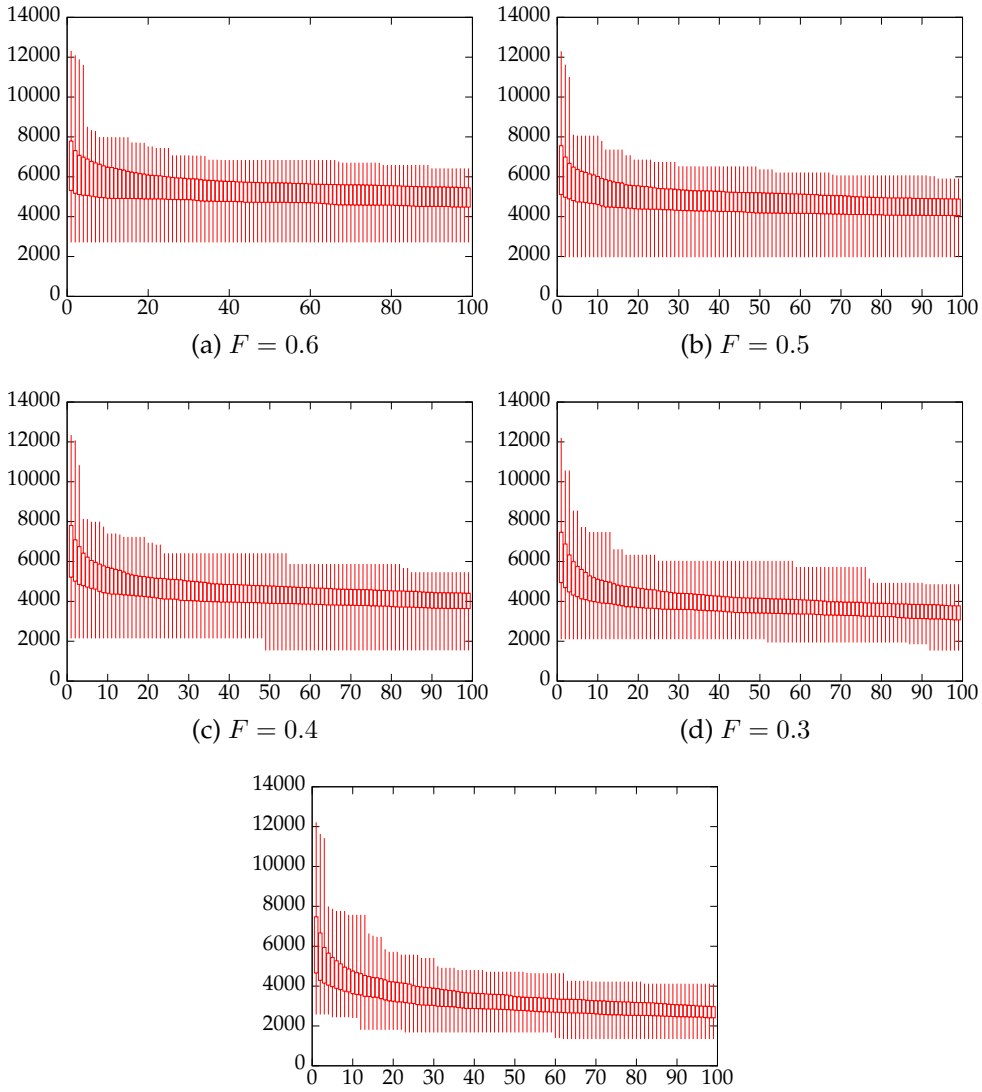


(a) $CR = 0.7$ (b) $CR = 0.8$

(c) $CR = 0.9$ (d) $CR = 0.95$

(e) $CR = 0.99$

Obrázek 36: Diferenciální evoluce pro různé hodnoty CR ($DE/rand/1/bin$, $PopSize = 100$, $F = 0.4$)

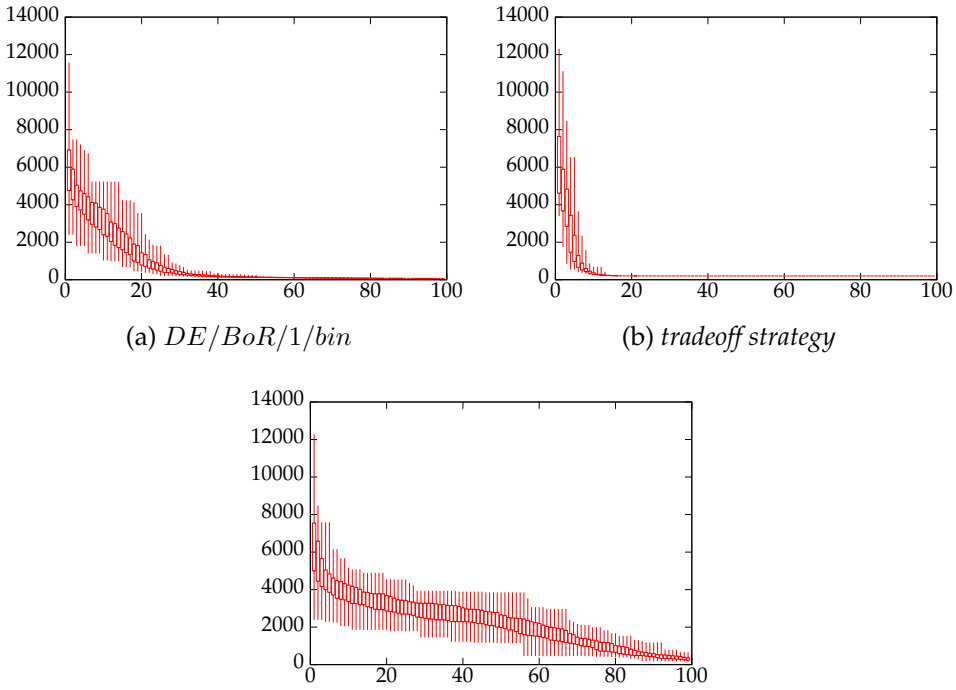


(a) $F = 0.6$ (b) $F = 0.5$

(c) $F = 0.4$ (d) $F = 0.3$

(e) $F = 0.2$

Obrázek 37: Diferenciální evoluce pro různé hodnoty F ($DE/rand/1/bin$, $PopSize = 100$, $CR = 0.8$)

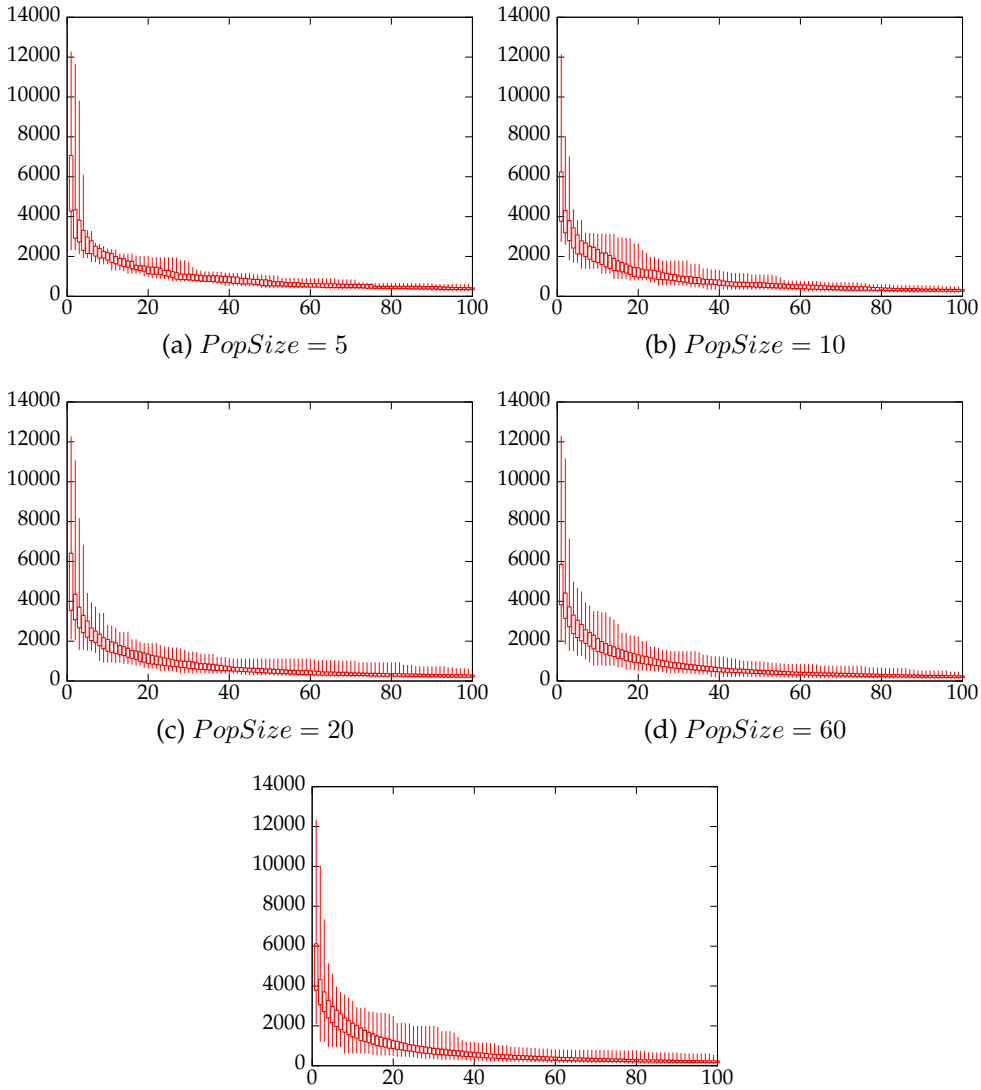


(a) $DE/BoR/1/bin$ (b) *tradeoff strategy*

(c) $DE/rand/1/bin$

Obrázek 38: Diferenciální evoluce pro různé typy výběru $r0$
($PopSize = 100$, $CR = 0.95$, $F = 0.2$)

A.2 SOMA

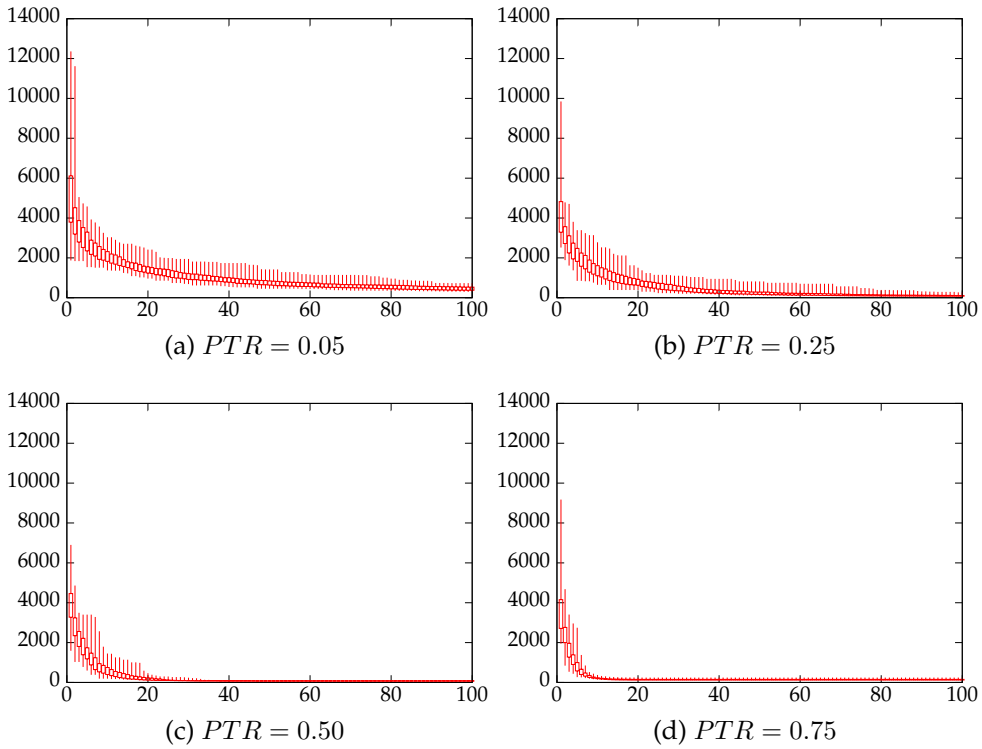


(a) $PopSize = 5$ (b) $PopSize = 10$

(c) $PopSize = 20$ (d) $PopSize = 60$

(e) $PopSize = 100$

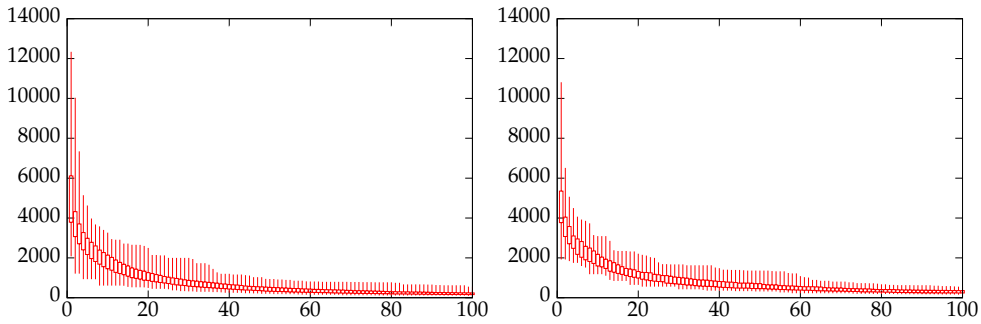
Obrázek 39: SOMA pro různé velikosti populace
($PathLength = 1.1$, $Step = 0.11$, $PTR = 0.1$)



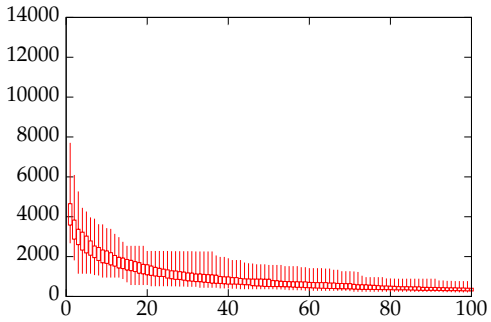
(a) $PTR = 0.05$ (b) $PTR = 0.25$

(c) $PTR = 0.50$ (d) $PTR = 0.75$

Obrázek 40: SOMA pro různé hodnoty PTR
($PopSize = 100$, $PathLength = 1.1$, $Step = 0.11$)

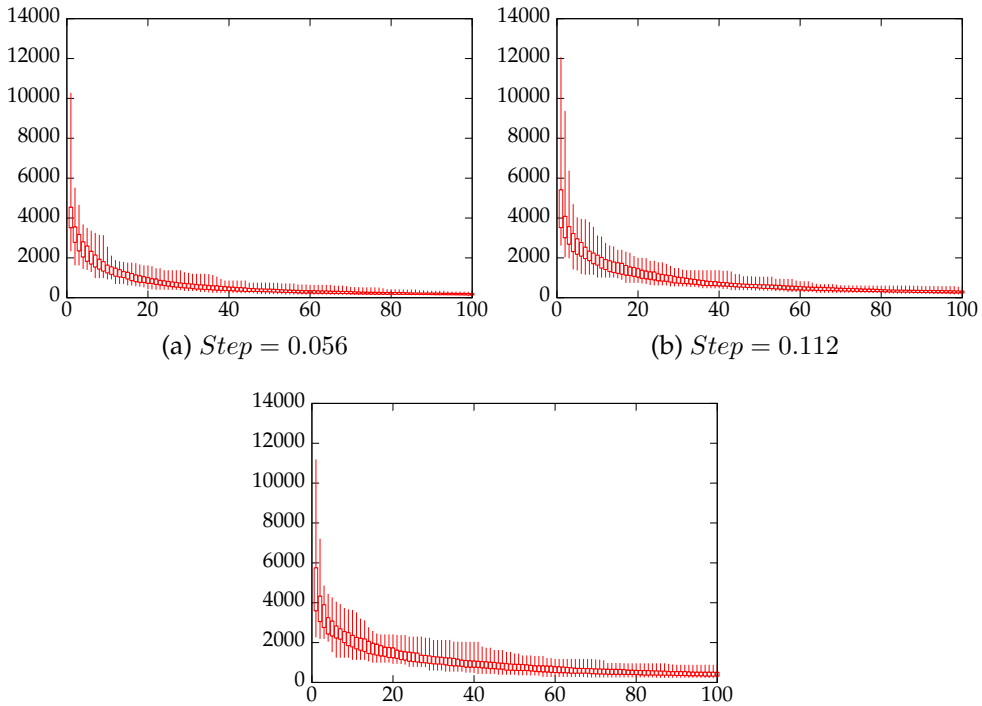


(a) $PathLength = 1.1$ (b) $PathLength = 1.5$



(c) $PathLength = 2.0$

Obrázek 41: SOMA pro různé hodnoty $PathLength$
($PopSize = 100$, $Step = 0.11$, $PTR = 0.1$)

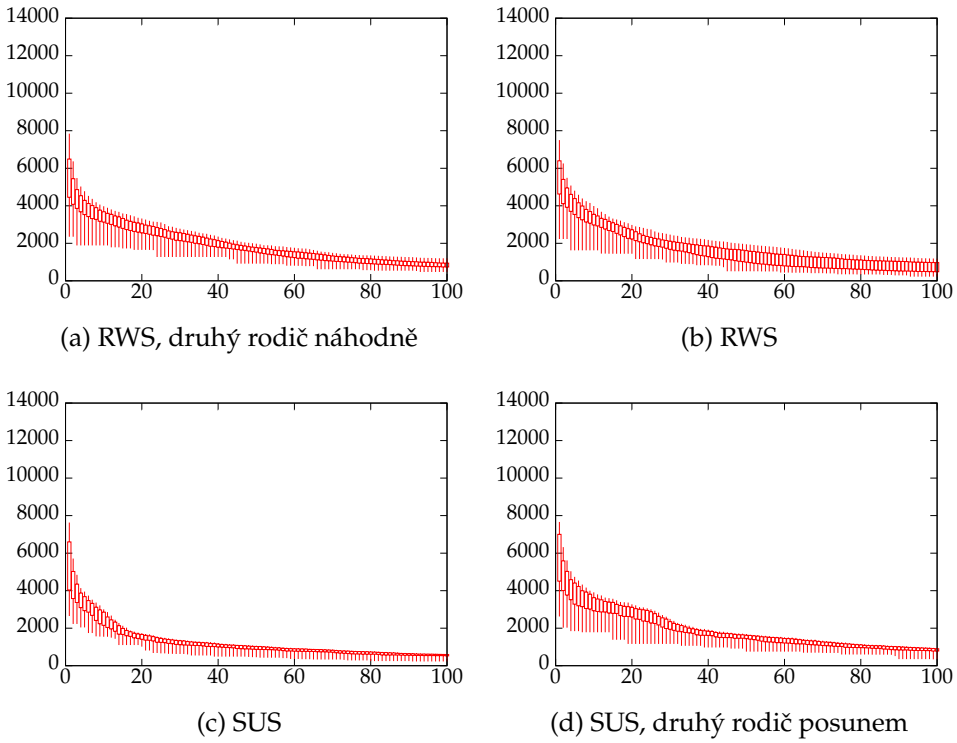


(a) $Step = 0.056$ (b) $Step = 0.112$

(c) $Step = 0.168$

Obrázek 42: SOMA pro různé hodnoty $Step$
($PopSize = 100$, $PathLength = 1.1$, $PTR = 0.1$)

A.3 Genetický algoritmus, binární reprezentace



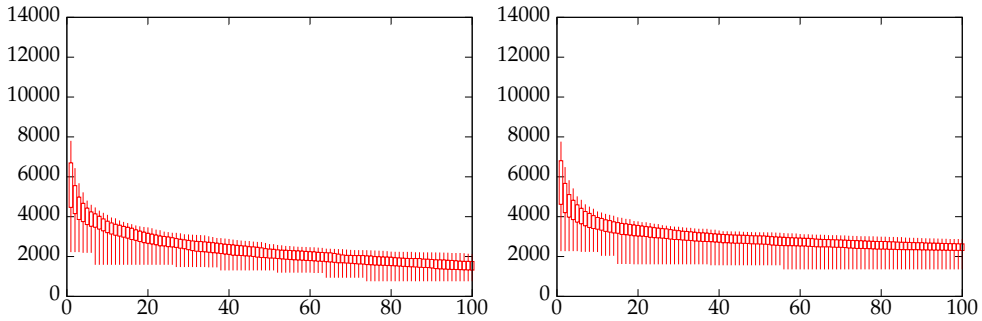
(a) RWS, druhý rodič náhodně

(b) RWS

(c) SUS

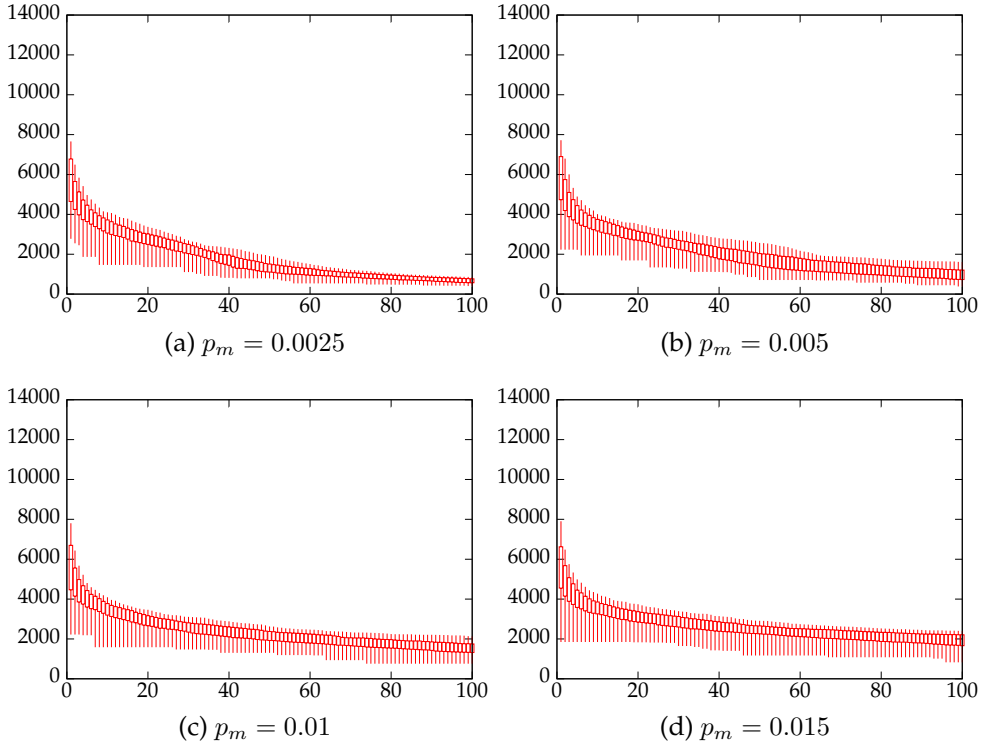
(d) SUS, druhý rodič posunem

Obrázek 43: Různé varianty výběru rodičů genetického algoritmu ($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$)



(a) One-point crossover (b) Uniform crossover

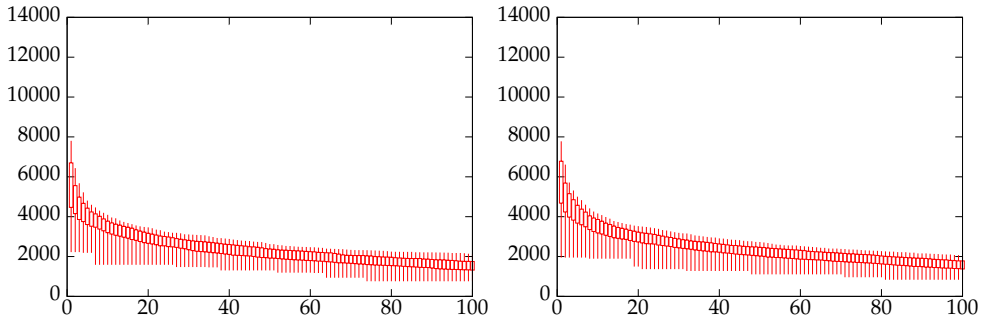
Obrázek 44: Různé varianty křížení v genetickém algoritmu
($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$)



(a) $p_m = 0.0025$ (b) $p_m = 0.005$

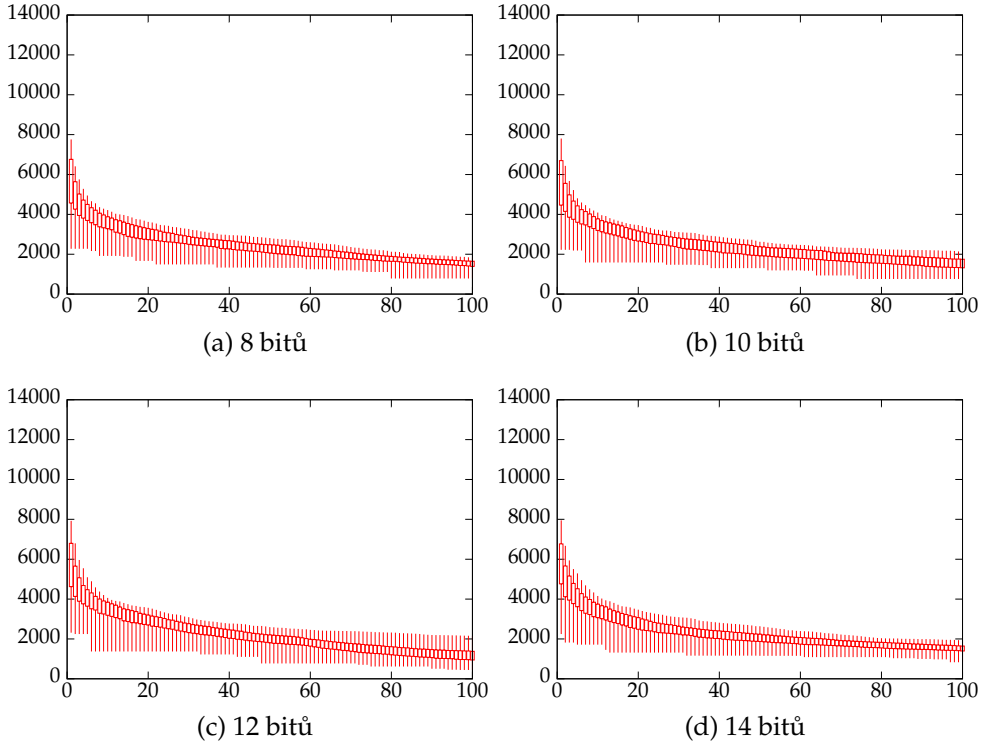
(c) $p_m = 0.01$ (d) $p_m = 0.015$

Obrázek 45: Genetický algoritmus pro různé hodnoty p_m
($PopSize = 100$, $p_{xo} = 0.95$, RWS)



(a) Binární reprezentace (b) Grayovým kódováním

Obrázek 46: Různé varianty binární reprezentace jedinců v genetickém algoritmu ($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$, RWS)

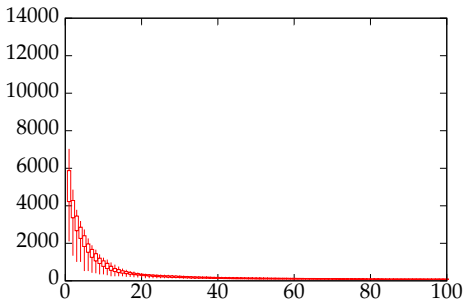


(a) 8 bitů (b) 10 bitů

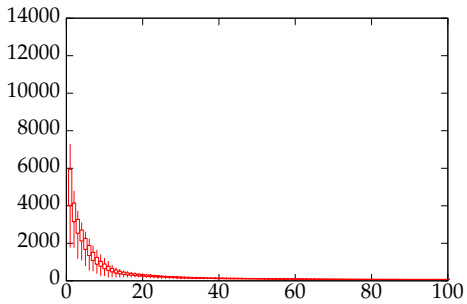
(c) 12 bitů (d) 14 bitů

Obrázek 47: Různé hodnoty počtu bitů reprezentujících číslo v genetickém algoritmu ($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$, RWS)

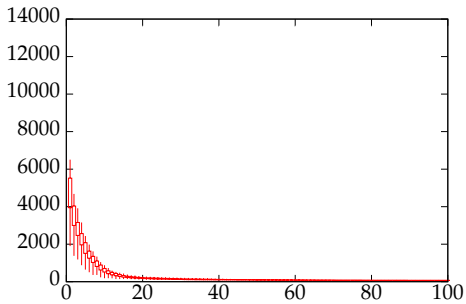
A.4 Genetický algoritmus, reprezentace reálnými čísly



(a) $p_{xo} = 0.6$

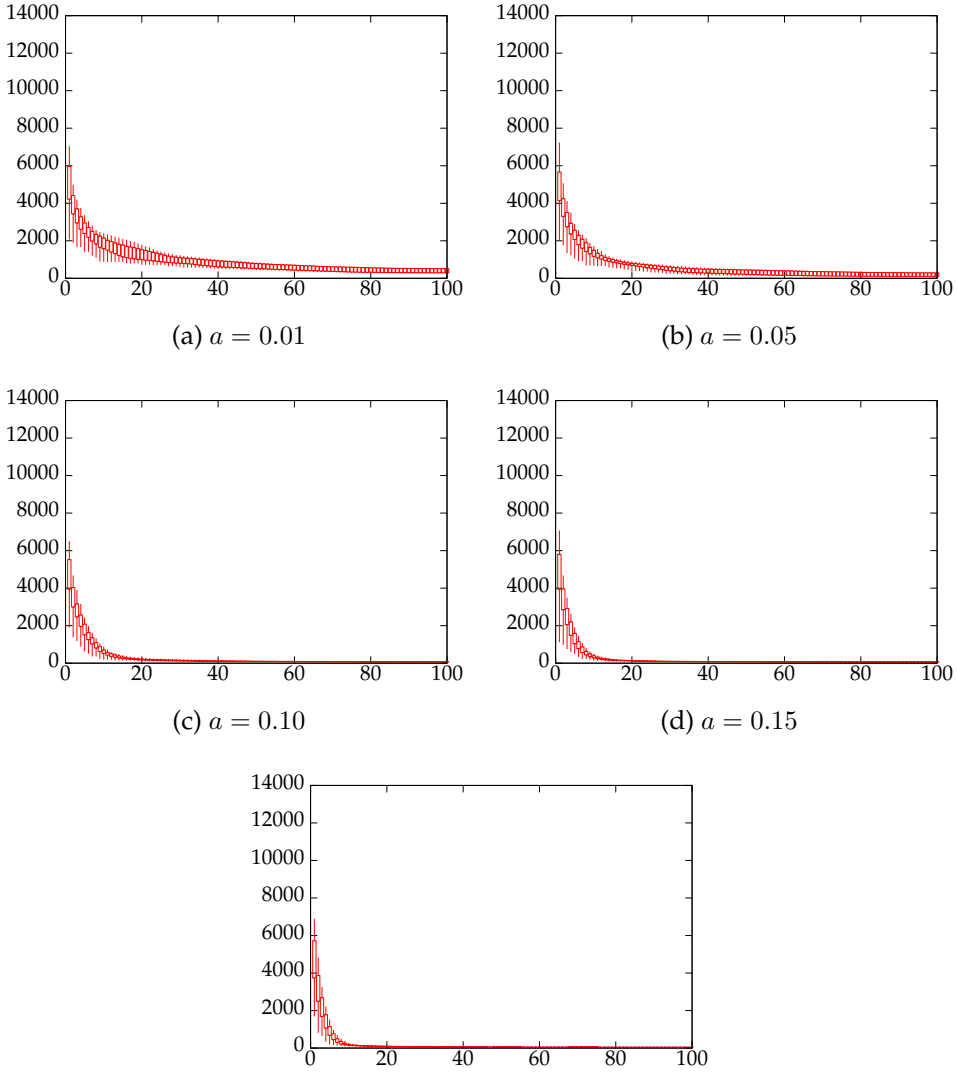


(b) $p_{xo} = 0.8$



(c) $p_{xo} = 0.95$

Obrázek 48: Genetický algoritmus pro různé hodnoty p_{xo} ($PopSize = 100$, $p_m = 0.01$, $a = 0.1$, $b = 2.0$)



(a) $a = 0.01$

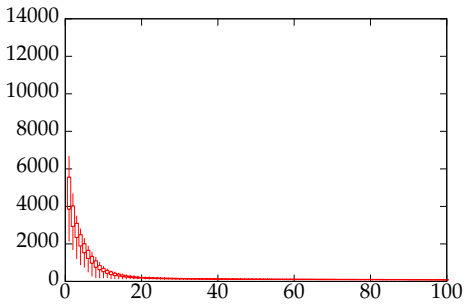
(b) $a = 0.05$

(c) $a = 0.10$

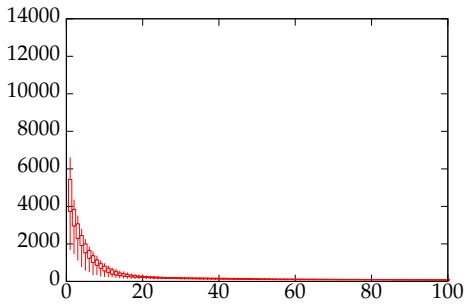
(d) $a = 0.15$

(e) $a = 0.20$

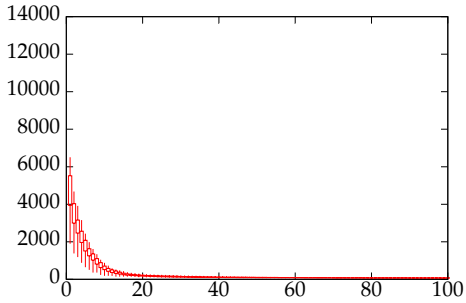
Obrázek 49: Genetický algoritmus pro různé hodnoty a
($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$, $b = 2.0$)



(a) $b = 0.5$

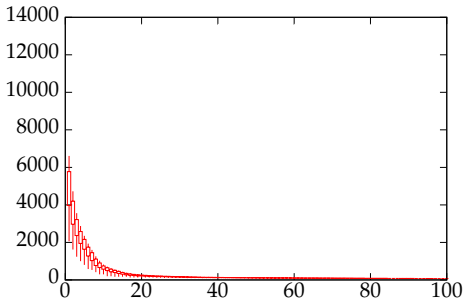


(b) $b = 1.0$

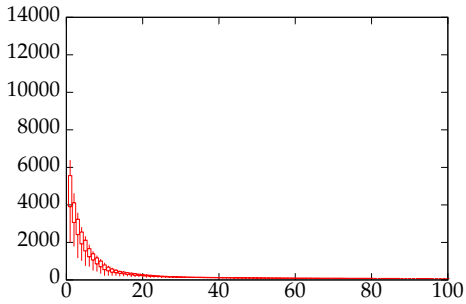


(c) $b = 2.0$

Obrázek 50: Genetický algoritmus pro různé hodnoty b
($PopSize = 100$, $p_{xo} = 0.95$, $p_m = 0.01$, $a = 0.1$)

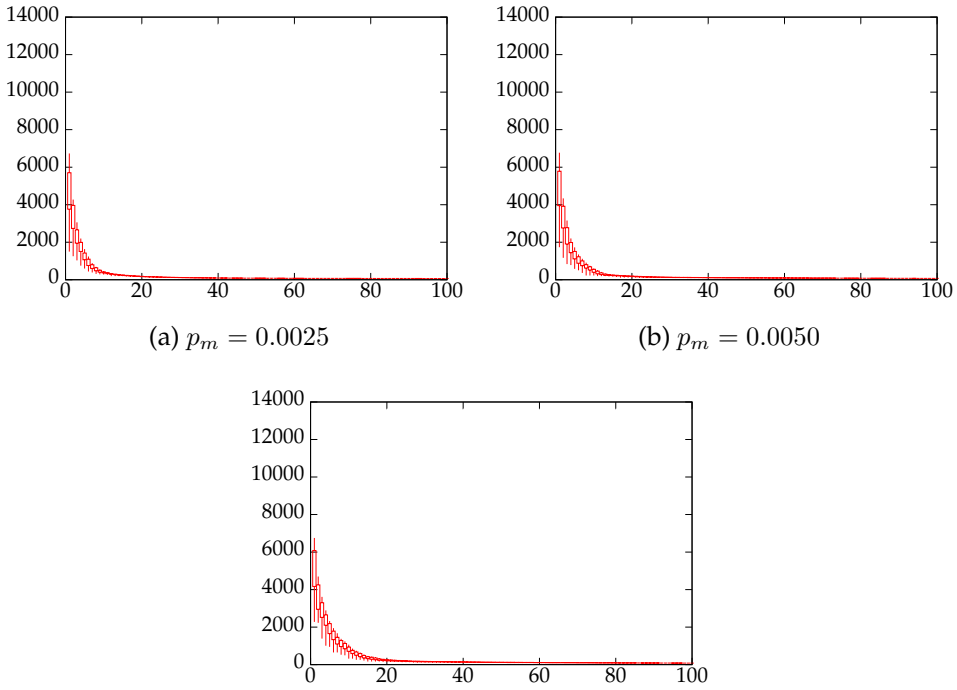


(a) RWS



(b) SUS

Obrázek 51: Genetický algoritmus pro různé metody selekce rodičů
($PopSize = 200$, $p_{xo} = 0.9$, $p_m = 0.0125$, $a = 0.1$, $b = 1.0$)

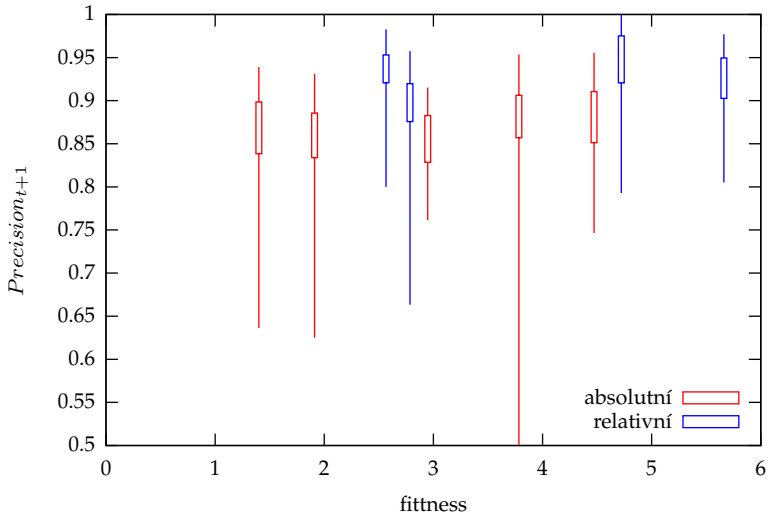


(a) $p_m = 0.0025$ (b) $p_m = 0.0050$

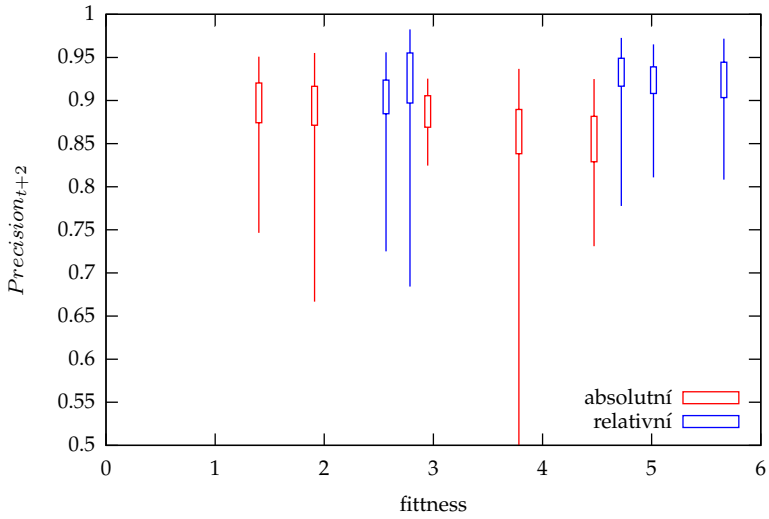
(c) $p_m = 0.0150$

Obrázek 52: Genetický algoritmus pro různé hodnoty p_m
($PopSize = 100$, $p_{xo} = 0.9$, $a = 0.1$, $b = 1.0$)

B Porovnání modelů

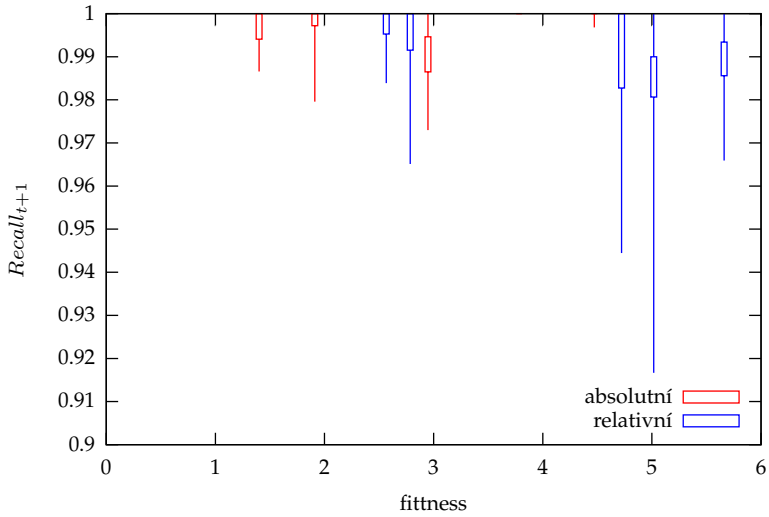


(a) $Precision_{t+1}$

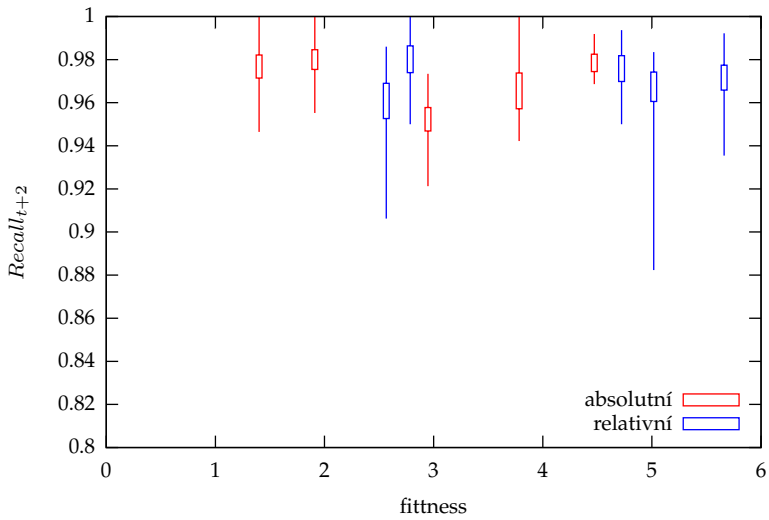


(b) $Precision_{t+2}$

Obrázek 53: $Precision$ pro model s absolutními a relativními hodnotami okolních bodů pro časy $t + 1$ a $t + 2$



(a) $Recall_{t+1}$



(b) $Recall_{t+2}$

Obrázek 54: $Recall$ pro model s absolutními a relativními hodnotami okolních bodů pro časy $t + 1$ a $t + 2$